Modelado dinámico y diseño de estrategia de control mediante estimadores para el vuelo autónomo de un quadrotor.

Autor: David Melero Cazorla

Directores: Antonio Giménez Fernández y José Luis Guzmán Sánchez

13 de septiembre de 2012

A todos lo que han ayudado
en la realización de este proyecto.

Índice

Parte I

Interés y objetivos

1. Interés

El término UAV, siglas en inglés de Unmanned Aerial Vehicle, se utiliza para definir a los vehículos aéreos sin tripulación humana a bordo. Los UAV se han convertido recientemente en una plataforma viable para tareas de supervivencia y exploración donde la presencia humana es peligrosa, imposible o inadecuada.

Los quadrotors son unos vehículos aéreos de cuatro rotores que poseen la capacidad de despegue y aterrizaje vertical (VTOL). Su simple diseño mecánico, su gran maniobrabilidad y la capacidad VTOL los hacen adecuados para que se usen como vehículos UAV en la vida real, por ejemplo es posible observar en equipos de policía de los Estados Unidos y Canadá que han usado los quadrotors de forma exitosa como equipamiento en tareas de supervivencia.

Además, en el ámbito de la investigación, los quadrotors se utilizan como plataforma para probar técnicas de control avanzado. Pese a su simple diseño mecánico, el modelo dinámico que define el comportamiento de los quadrotors es complejo debido a que se trata de un modelo no lineal, es por ello que se utilizan como banco de ensayos para el estudio sistemas de control y observadores no lineales.

El filtro de Kalman ha sido uno de los descubrimientos que ha permitido el avance en el campo del control de sistemas dinámicos complejos. Para controlar un sistema dinámico, primero se tiene que conocer y capturar su comportamiento. En estas aplicaciones, no siempre es posible o deseable medir todas las variables a controlar, el filtro de Kalman proporciona un medio para inferir la información desconocida a partir de mediciones indirectas y ruidosas.

2. Objetivos

Este proyecto se considera continuación del Proyecto de Fin de Carrera de Rubén Jiménez Gajate, *Diseño de un sistema de control para un quadrotor* [18], en el cual se habilitó la plataforma acual para diseñar e implementar diversos tipos de sistemas de control sobre el quadrotor. En el presente proyecto se plantea el modelado dinámico del quadrotor así como el diseño e implementación de un sistema de control que permita un vuelo estático autónomo. Además, se realizará el modelado y la simulación en MATLAB/Simulink de un algoritmo de control que permita el seguimiento de trayectorias previamente definidas.

Por tanto los objetivos de este proyecto se pueden resumir en los siguientes puntos:

- Modelado dinámico del quadrotor y comprobación de su validez mediante simulación.

- Diseñar un filtro de Kalman que permita obtener la orientación a partir de las mediciones de los sensores con los que está equipado el quadrotor, acelerómetro y giroscopio triaxial.
- Diseño de un controlador tipo PID que permita un vuelo estático, autónomo y estable frente a perturbaciones externas.
- Implementación de dicho algoritmo de control en la plataforma electrónica Arduino con la que está dotado el quadrotor.
- Diseño y simulación de un algoritmo de control que permita al quadrotor el seguimiento de trayectorias previamente definidas.

Parte II

Revisión bibliográfica

3. Vehículos aéreos no tripulados

Un vehículo aéreo no tripulado, UAV por las siglas en inglés (*Unmanned Aerial Vehicle*), es una aeronave que es capaz de realizar una misión sin tripulación humana a bordo. Su vuelo puede ser dirigido por controladores autónomos incorporados en la aeronave, o mediante control remoto desde tierra u otro vehículo.

Puesto que la anterior definición no excluye el telecontrol de la aeronave, cabe definir vehículo aéreo autónomo, AAV (*Autonomous Aerial Vehicle*) como aquél capaz de desarrollar la misión sin necesidad de intervención humana. En este caso cabría la posibilidad de que la aeronave transportara personal, no dedicado a la misión, pero esta posibilidad, similar al piloto automático con el que cuentan la mayor parte de los aviones actuales, queda fuera del contexto en el que los sistemas considerados operan.

El principal uso de los UAV ha sido en tareas militares, realizando tanto misiones de reconocimiento como de ataque. Pero los UAV también se emplean en un pequeño pero creciente número de aplicaciones civiles; relacionadas con la teledetección en su mayor parte, por ejemplo en vigilancia y seguridad civil (lucha contra incendios o la supervisión de oleoductos), y en menor grado con el transporte. En general, los UAV suelen ser preferidos para misiones que son demasiado "monótonas, sucias o peligrosas" para aeronaves tripuladas [4, 55].

3.1. Evolución histórica de los vehículos aéreos no tripulados

El primer intento de vehículo aéreo no tripulado fue el "*Aerial Target*" de A. M. Low, el considerado padre de los sistemas de guiado por radio, en 1916. Un año antes, en 1915, Nikola Tesla describe una flota de aviones no tripulados de combate aéreo. Le siguieron una serie de avances en materia de aviones a control remoto, incluyendo el avión automático Hewitt-Sperry, durante y tras la Primera Guerra Mundial, entre ellos se encuentra el primer vehículo a escala pilotado a distancia, desarrollado por la estrella de cine y aficionado al aeromodelismo Reginald Denny en 1935. Se realizaron más avances durante la carrera técnológica de la Segunda Guerra Mundial, los cuales fueron utilizados tanto como para entrenar a los artilleros antiaéreos como para las misiones de ataque. La Alemania nazi también produjo y utilizó varios aviones UAV durante el curso de la Segunda Guerra Mundial. Los motores a reacción llegarían después de la Segunda Guerra Mundial, en modelos como el Teledyne Ryan Firebee I de 1951, o los de otras compañías que también entraron en el juego como Beechcraft con su Modelo 1001 para la Marina de los Estados Unidos en 1955. Sin embargo, hasta la guerra de Vietnam eran poco más de aviones a control remoto.

El nacimiento de los UAV en EE.UU. (llamados RPV en el momento) se inició en 1959, cuando los oficiales de las Fuerzas Aéreas estadounidenses (USAF), preocupados por

Figura 1: AQM-34 usado en misiones de reconocimiento durante la guerra de Vietnam por la USAF

la pérdida de los pilotos sobre territorio hostil, comenzaron la planificación para el uso de los vuelos no tripulados (ver figura 1).

Este plan se intensificó cuando Francis Gary Powers y su U-2 "secreto" fueron derribados sobre la Unión Soviética en 1960. En pocos días, el programa de alto secreto sobre los UAV se lanzó bajo el nombre en clave de “Red Wagon”. El choque el 2 de agosto y 4 de agosto de 1964, en el Golfo de Tonkin entre las unidades navales de los EE.UU. y la armada de Vietnam del Norte inició el uso por parte de los americanos de los UAV en sus misiones de combate iniciales de la guerra de Vietnam. Cuando la “China comunista” mostró fotografías de los UAV derribados de EE.UU. a través de fotos al mundo, la respuesta oficial de EE.UU. fue: “sin comentarios”.

El 26 de febrero de 1973 los militares de EE.UU. confirmaron oficialmente que habían estado utilizando UAVs en el sudeste asiático (Vietnam). Mientras que más de 5.000 aviadores estadounidenses habían resultado muertos y más 1.000 más desaparecieron en acción o fueron capturados; el 100º Escuadrón de Reconocimiento Estratégico de la USAF había realizado unas 3.435 misiones con UAV durante la guerra, con un costo de unos 554 vehículos aéreos no tripulados perdidos por causas variadas. En palabras del General de la USAF George S. Brown “la única razón por la que necesitamos (UAVs) es que no queremos gastar innecesariamente el hombre de la cabina". Más tarde, ese mismo año, el general John C. Meyer, comandante en jefe del Comando, Aéreo Estratégico, declaró: "dejamos que el avión no tripulado haga los vuelos con riesgo alto... la tasa de pérdida es alta, pero estamos dispuestos a arriesgar más de ellos... salvan vidas."

Durante la guerra en 1973 de Yom Kippur, las baterías sirias de misiles en Líbano causaron graves daños a los aviones de combate israelíes. Como resultado, Israel desarrolló el primer UAV moderno. Israel fue pionera en el uso de vehículos aéreos no tripulados para vigilancia en tiempo real, la guerra electrónica y señuelos. Las imágenes y los señuelo de radar proporcionados por estos UAV ayudaron a Israel a neutralizar por completo las defensas antiaéreas sirias en el inicio de la Guerra de Líbano en 1982,

Figura 2: vehículo aéreo no tripulado General Atomic MQ-1 Predator

como resultado no hubo pilotos derribados.

Con la evolución y la miniaturización de las tecnologías aplicadas a los UAV, en los años 80 y 90, creció el interés por ellos entre los altos cargos militares de los EE.UU.. En los años 90 el Departamento de Defensa de los EE.UU. dio un contrato a la corporación AAI junto con la empresa israelí Mazlat. La Marina estadounidense adquirió el UAV AAI Pioneer desarrollado fruto de este contrato, este tipo de aeronave no tripulada se encuentra aun en uso. Muchos de estos Pioneer y nuevos UAV desarrollados se usaron en la Guerra de Golfo en 1991. Las primeras generaciones tuvieron como función labores de vigilancia, pero algunos de ellos estaban armados, como es el caso del Genera Atomics MQ-1 Predator (figura 2) que portaba misiles aire-tierra AGM-114 Hellfire [55].

3.2. Clasificación de los UAV

A la hora de establecer una clasificación de los UAV es posible atender a diferentes criterios. Tal vez el más simple sea el que se basa en el tipo de la aeronave del UAV. De acuerdo a este pueden distinguirse aquellas aeronaves de despegue vertical de las que no lo son, estando dentro de las primeras las de ala rotativa o hélice (helicópteros y quadrotors), los de ala flexible (parapentes y ala delta), y los autosustentados (dirigibles y globos). Dentro de los de despegue no vertical, se encuentran los de ala fija (aeroplanos). La figura 3, muestra los diferentes tipos de aeronave utilizados en los UAV.

Las prestaciones y, en consecuencia, las aplicaciones varían mucho de un tipo de aeronave a otra, cubriendo cada uno de ellos un espectro diferente. En la tabla 1 se recogen algunas de las principales características de las principales aeronaves utilizadas para UAV [4].

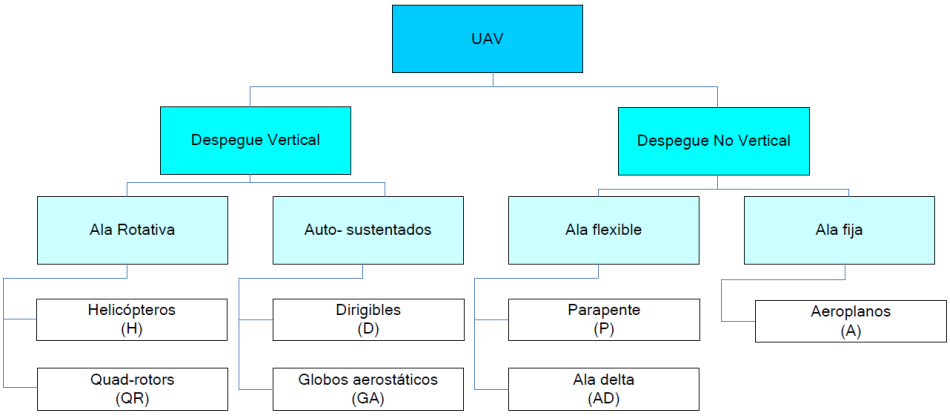

Figura 3: clasificación de los UAV en función de la aeronave

Tabla 1: características de los principales tipos de aeronaves para UAV

Característica	Helicóptero	Aeroplano	Dirigible	Quadrotor
Capacidad de vuelo estacionario	***		****	***
Velocidad de desplazamiento	***	****	*	**
Maniobrabilidad	***	*	*	****
Autonomía de vuelo	**	***	****	*
Resistencia a perturbaciones externas	**	****	*	**
Auto Estabilidad	*	***	****	**
Capacidad de vuelos verticales	****	*	**	****
Capacidad de carga	***	****	*	**
Capacidad de vuelo en interiores	**	*	***	****
Techo de vuelo	**	****	***	*

Por último, otro tipo de clasificación, es la que se realiza por su función. Los vehículos aéreos no tripulados se pueden clasificar por su función en alguna de las seis categorías siguientes:

- Blanco y señuelo, proporcionan a la artillería terrestre y aérea un blanco que simula un avión o misil enemigo.
- Reconocimiento, proporcionan información en el campo de batalla, en escenarios de emergencia o en ambientes de alta toxicidad química o radiológica.
- Combate, proporcionan capacidad de ataque en misiones de alto riesgo.
- Logística, son diseñados específicamente para la carga y operaciones de logística.
- Investigación y desarrollo, usados para desarrollar la futura tecnología de los UAV.
- Civil y comercial, vehículos aéreos no tripulados diseñados específicamente para aplicaciones civiles y comerciales. Cartografía, gestión de cultivos, prospección de yacimientos geológicos, gestión del medio ambiente, hidrología, control urbanístico, etc.

3.3. Quadrotor

Un quadrotor es una aeronave que se eleva y se desplaza por la acción de cuatro rotores instalados en los extremos de una estructura en forma de cruz. De los cuatro rotores, dos giran en sentido horario, y los otros dos en sentido antihorario. Esta configuración permite al helicóptero realizar cualquier maniobra tal y como se explicará posteriormente.

Una de las principales ventajas que se atribuyen a los quadrotors son la simplicidad mecánica del vehículo, el paso de las hélices de los rotores es fijo, cosa que no ocurre en los helicópteros donde se requiere un mecanismo que permita la variación del ángulo de ataque. El control del movimiento se realiza al generar momentos debido a diferencias de empuje entre los diferentes rotores. En los helicópteros convencionales, la velocidad de giro de las hélices suele ser constante y el control del movimiento se realiza mediante la variación de los ángulos de ataque de las palas.

Otra ventaja es el empleo de motores eléctricos en lugar de motores de combustión. Esto es especialmente interesante si se van a realizar vuelos en interior donde existe la posibilidad de que se contamine el aire por los productos de la combustión. Sin embargo, existe un inconveniente, la utilización de motores eléctricos limita el tiempo de funcionamiento, ya que las baterías que alimentan los motores aportan autonomía no superior a los 15 o 20 minutos [28].

La alta maniobrabilidad es otra de las características de los quadrotors. En algunos de los artículos del grupo de investigación del *Flying Machine Arena* del ETH de Zurich se habla de estrategias de aprendizaje para la ejecución de múltiples "*flips*" [24], así como de maniobras acrobáticas adaptativas en bucle abierto [25] (ver figura 4). Los

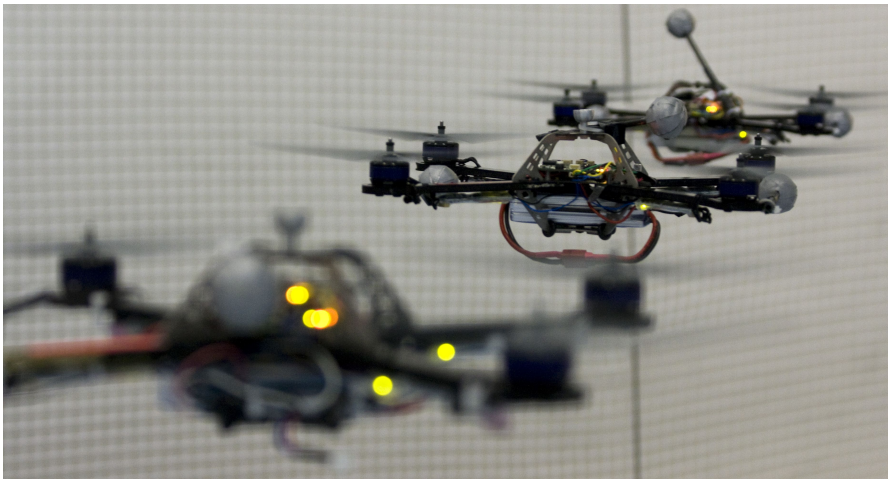

Figura 4: tres quadrotors volando en el interior del *Flying Machine Arena* del ETH

vídeos publicados en internet por este grupo de investigación han dado la vuelta al mundo gracias a las redes sociales ayudando a que aumente el interés del público general así como de otros grupos de investigación en materia de quadrotors. [42]

Como gran desventaja se suele citar la dificultad de establecer un sistema de control, especialmente para aquellos quadrotors de dimensiones reducidas. Esto se debe al hecho de que las fuerzas y pares necesarios para controlar el sistema son conseguidos no sólo a través de efectos aerodinámicos, sino también a través del acoplamiento de la dinámica de los cuatro rotores [28].

3.3.1. Aplicaciones actuales

A lo largo de los últimos años, se ha producido un rápido aumento del número de proyectos que trabajan con quadrotors. La mayoría de quadrotors de estos proyectos pertenecen a la categoría de mini UAV, donde su peso es menor de 10 Kg.

Algunos de ellos se utilizan como juguetes manejados por radiocontrol, como es el caso del AR.Drone de Parrot (figura 5) que se maneja desde un dispositivo móvil equipado con el sistema operativo iOS o Android. Otros prototipos más complejos han comenzado a emplearse en el campo civil y militar, posicionando al quadrotor como uno de los modelos de mayor desarrollo en el campo de los UAV. Prueba de ello es la adquisición por los cuerpos de seguridad de UAV del tipo quadrotor para realizar labores de vigilancia, principalmente por la capacidad que tienen estas aeronaves de alcanzar espacios de difícil acceso [28, 38].

3.3.2. Maniobrabilidad

Cada rotor produce un empuje y un momento sobre su eje de rotación, así como una pequeña fuerza de arrastre de sentido opuesto a la dirección de vuelo del vehículo.

Figura 5: quadrotor comercial Parrot AR.Drone

Si todos los rotores giran a la misma velocidad angular, con los motores uno y tres rotando en sentido horario y los motores dos y cuatro en sentido antihorario, el par aerodinámico neto, y, por tanto, la aceleración angular sobre el eje de guiñada es exactamente cero, esto implica que no es necesario un rotor para controlar la guiñada como ocurre en los helicópteros (ver figura 6). El movimiento de guiñada es inducido desequilibrando el balance del par aerodinámico, por ejemplo reduciendo la velocidad de los rotores que giran en sentido antihorario.

Las aceleraciones angulares en los ejes de alabeo y cabeceo se pueden generar sin producir impacto en el eje de guiñada. Cada par de hélices rotando en el mismo sentido controlan un eje, alabeo y cabeceo. Incrementando el empuje en un rotor mientras se reduce en el otro se mantendrá el balance de par necesario para mantener la estabilidad de la guiñada a la vez que se induce un par sobre el eje de alabeo o cabeceo, ver figuras 6 y 7. De este modo es posible mover el quadrotor en cualquier dirección utilizando hélices de paso fijo. La traslación se consigue manteniendo un ángulo de alabeo o cabeceo distinto de cero [52].

4. Filtro de Kalman

Teóricamente el filtro de Kalman es un estimador para lo que se conoce como problema lineal cuadrático, que consiste en la estimación del estado actual de un sistema dinámico perturbado por ruido blanco, usando mediciones relacionadas linealmente

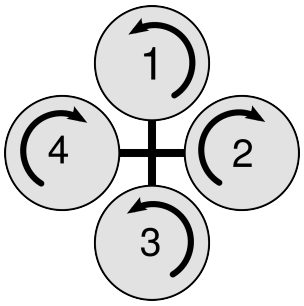

Figura 6: sentido de giro de los rotores del quadrotor

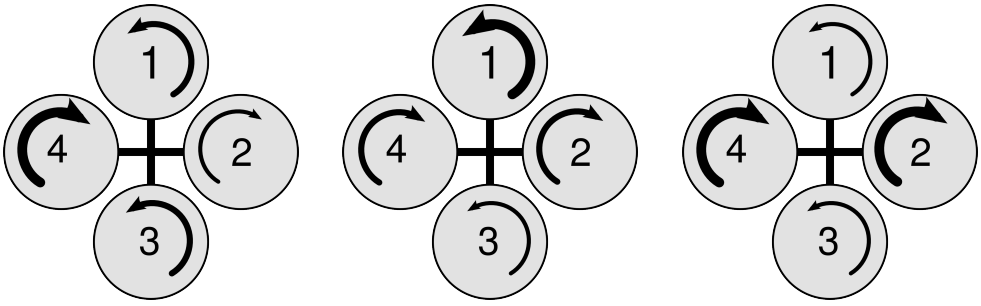

Figura 7: cambio en la velocidad de los rotores para ejecutar una rotación positiva en alabeo, cabeceo y guiñada, respectivamente

pero corrompidas con ruido blanco aditivo. El estimador resultante es estadísticamente óptimo con respecto a cualquier función cuadrática de estimación del error.

En la práctica, ha sido uno de los grandes descubrimientos en la historia de la teoría de la estimación estadística y ha posibilitado grandes descubrimientos en el siglo XX. De ha hecho tan indispensable como el silicio en la fabricación de componentes electrónicos. Su más inmediata aplicación ha sido el control de sistemas dinámicos complejos como sistemas de producción en continuo, aeronaves, barcos o naves espaciales. Para controlar un sistema dinámico, primero hay que conocer y capturar su comportamiento. En estas aplicaciones, no siempre es posible o deseable medir cada una de las variables a controlar, y es el filtro de Kalman quien provee un medio para inferir la información desconocida de mediciones indirectas y ruidosas. El filtro de Kalman es usado también para predecir el futuro comportamiento de sistemas dinámicos que probablemente no se van a poder controlar, como el curso de los ríos durante una inundación, la trayectoria de cuerpos celestes o los precios de las acciones en los mercados de valores.

Puede parecer extraño que se aplique el término "filtro" a un estimador. Más comúnmente, un filtro es un instrumento físico que separa la parte no deseada de una mezcla. Originalmente, un filtro resolvía el problema de separar componentes no deseados de una mezcla. En la era de las radios de cristal y los tubos de vacío, el término fue aplicado al circuito analógico que filtraba las señales electrónicas. Estas señales son una mezcla de componentes de diferentes frecuencias, y estos instrumentos físicos atenuaban las frecuencias no deseadas.

Este concepto fue extendido en los años 30 y 40 al de separar "señales" del "ruido", ambas caracterizadas por su densidad espectral de potencia. Kolmogorov y Wiener usaron esta característica estadística de las distribuciones de probabilidad para obtener una estimación óptima de la señal, dada la suma de señal y ruido.

Con el filtro de Kalman el término filtrado asumió un significado que va más allá de la idea original de separar componentes de una mezcla. Ha llegado a incluir la solución de un problema de inversión, en el cual se conoce como representar las variables medidas como función de las principales variables de interés. En esencia, invierte la relación funcional y estima las variables independientes como funciones inversas de las variables dependientes (medidas). Estas variables de interés también pueden ser dinámicas, con dinámicas que son sólo parcialmente predecibles [16, 47].

4.1. Filtro de Wiener

En los primeros años de la Segunda Guerra Mundial, Wiener estaba involucrado en un proyecto militar para el desarrollo de un sistema antiaéreo automático usando la información de un radar. Como la velocidad de la aeronave no es insignificante en comparación con la de la bala, el sistema tenía que "disparar en el futuro", es decir, el controlador tenía que predecir la trayectoria futura usando la información del seguimiento realizado por el radar.

Wiener derivó la solución para la predicción del error por mínimos cuadrados en términos de las funciones de autocorrelación de la señal y el ruido. La solución se realiza

mediante un operador integral que puede ser implementado mediante circuitos analógicos, dadas ciertas limitaciones en la regularidad de las funciones de autocorrelación o, de forma equivalente, sus transfomadas de Fourier.

Una solución análoga del estimador lineal óptimo para sistemas en tiempo discreto fue publicada por A. N. Kolmogorov en 1941, cuando Wiener estaba terminando su trabajo en el estimador en tiempo continuo [16].

4.2. Filtro de Kalman

Desarrollado por R. E. Kalman en 1958, surgió de la idea de aplicar la notación de las variables de estado al filtro Wiener. Con la suposición adicional de dimensiones finitas, fue capaz de derivar del filtro Wiener a lo que hoy conocemos como filtro de Kalman. Al usar la notación del espacio de estados los conocimientos necesarios para su derivación eran menores [16].

El filtro de Kalman aborda el problema general de intentar estimar el estado $x \in \Re^n$ de un proceso controlado en tiempo discreto que es gobernado por una ecuación diferencial lineal estocástica del tipo

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \tag{4.1}$$

con una medición $z \in \Re^m$ que es

$$z_k = Hx_k + v_k. \tag{4.2}$$

La variables aleatorias w_k y v_k representan el ruido del proceso y de la medición respectivamente. Se supone que son independientes una de la otra, blancas, y con una distribución de probabilidad normal.

$$p(w) \sim N(0, Q) \tag{4.3}$$

$$p(v) \sim N(0, R) \tag{4.4}$$

En la práctica, la matriz de covarianza del ruido del proceso Q y la matriz de covarianza del ruido de las mediciones R pueden variar en cada paso de tiempo o medición, sin embargo se asume que son constantes.

La matriz A, de dimensiones $n \times n$, relaciona el estado en el etapa anterior $k-1$ con el estado en la etapa actual k, en ausencia de una función de entrada controladora o de ruido en el proceso. La matriz B, de dimensiones $n \times l$, relaciona la entrada de control $u \in \Re^l$ con el estado x. La matriz H, de dimensiones $m \times n$, en la ecuación de medición relaciona el estado con la medición z_k.

Definiéndose $\hat{x}_k^- \in \mathbb{R}^n$ como la estimación a priori del estado en una etapa k conociendo el proceso en el estado del proceso antes de ésta, y $\hat{x}_k \in \mathbb{R}^n$ como el estado

estimado a posteriori en la etapa k dada la medición z_k. Por tanto, se pueden definir los errores a priori y a posteriori como

$$e_k^- \equiv x_k - \hat{x}_k^-, \; y \tag{4.5}$$

$$e_k \equiv x_k - \hat{x}_k. \tag{4.6}$$

La covarianza del error estimada a priori es, por tanto,

$$P_k^- = E\left[e_k^- e_k^{-T}\right], \tag{4.7}$$

y la covarianza del error estimada a posteriori es

$$P_k = E\left[e_k e_k^T\right]. \tag{4.8}$$

Al deducir las ecuaciones del filtro de Kalman, se comienza con el objetivo de encontrar una ecuación que calcule una estimación del estado a posteriori $\hat{x}_k$ como una combinación lineal de la estimación a priori $\hat{x}_k^-$ y un promedio ponderado de la diferencia entre la actual medida z_k y una predicción de la medida $H\hat{x}_k^-$ como se muestra a continuación.

$$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right) \tag{4.9}$$

El origen se encuentra en la probabilidad del estado estimado a priori $\hat{x}_k^-$ condicionado a todas las mediciones anteriores z_k (regla de Bayes). El filtro de Kalman mantiene los dos primeros momentos de la distribución del estado,

$$E\left[x_k\right] = \hat{x}_k, \tag{4.10}$$

$$E\left[\left(x_k - \hat{x}_k\right)\left(x_k - \hat{x}_k\right)^T\right] = P_k. \tag{4.11}$$

El estado estimado a posteriori (4.9) refleja que la media (primer momento) de la distribución del estado están normalmente distribuidas si se reúnen las condiciones (4.3) y (4.4). La covarianza del error estimada a posteriori refleja la varianza de la distribución de estado (el segundo momento no centrado). En otras palabras,

$$p\left(x_k \mid z_k\right) \sim N\left(E\left[x_k\right], E\left[\left(x_k - \hat{x}_k\right)\left(x_k - \hat{x}_k\right)^T\right]\right) = N\left(\hat{x}_k, P_k\right) \tag{4.12}$$

La diferencia $\left(z_k - H\hat{x}_k^-\right)$ en (4.9) es conocida como innovación o el residuo. El residuo refleja la discrepancia entre la medida predicha $H\hat{x}_k^-$ y la medida actual z_k. Un residuo de cero significa que las dos están completamente de acuerdo.

La matriz K de dimensiones $n \times m$ de (4.9) es escogida para ser la ganancia o factor de mezcla que minimiza la covarianza del error a posteriori (4.8). Una de las formas de llegar a la minimización de K es

$$K_k = P_k^- H^T \left(H P_k^- H^T + R\right)^{-1} \tag{4.13}$$

$$= \frac{P_k^- H^T}{\left(H P_k^- H^T + R\right)}$$

Observando la ecuación 4.13 se puede ver cómo en el caso de que la covarianza del error de la medición R se aproxime a cero, la ganancia K le da más peso a ponderación del residuo. Específicamente,

$$\lim_{R_k \longrightarrow 0} K_k = H^{-1}. \tag{4.14}$$

En el caso contrario, cuando la covarianza del error a priori P_k^- se aproxima a cero, la ganancia K le da menos peso en la ponderación al residuo. Concretamente,

$$\lim_{P_k^- \longrightarrow 0} K_k = 0. \tag{4.15}$$

Otra forma de pensar en la ponderación realizada por K es que cuando la covarianza del error de la medición R se aproxima a cero, se confía más en la medición actual z_k, mientras que se confía menos en la medición estimada $H\hat{x}_k^-$. Por otro lado, cuando la covarianza del error estimado P_k^- se aproxime a cero, se confiará menos en la medición actual z_k y más en la medición estimada $H\hat{x}_k^-$ [16, 35].

4.2.1. Filtro de Kalman discreto

El filtro de Kalman estima el estado del proceso mediante el uso de una forma de control retroalimentado: el filtro estima el estado del proceso en algún momento y después obtiene una retroalimentación en forma de mediciones ruidosas. De esta forma las ecuaciones del filtro de Kalman se dividen en dos etapas: la etapa de predicción y la etapa de corrección.

La etapa de predicción es la responsable de proyectar hacia delante en el tiempo el estado actual y la covarianza del error estimada para obtener estimaciones a priori para la siguiente etapa. La etapa de corrección es la encargada de la retroalimentación, por ejemplo, incorporando una nueva medida en la estimación a priori para obtener una estimación a posteriori mejor.

Las ecuaciones de las etapas de predicción y corrección se muestran en las tablas 2 y 3.

Tabla 2: ecuaciones de predicción del filtro de Kalman discreto

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

$$P_k^- = AP_{k-1} + A^T + Q$$

Tabla 3: ecuaciones de corrección del filtro de Kalman discreto

$$K_k^- = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H\hat{x}_k^-\right)$$

$$P_k = \left(I - K_k H\right) P_k^-$$

Otra vez se puede ver cómo las ecuaciones de predicción de la tabla 2 proyectan el estado y la covarianza hacia delante de la etapa $k-1$ a la etapa k. Las matrices A y B provienen de (4.1) mientras que Q lo hace de (4.3).

La primera tarea durante la etapa de corrección es el cálculo de la ganancia de Kalman, K_k. El siguiente paso es actualizar las mediciones para obtener un nuevo z_k, y después, operar para generar una estimación a posteriori incorporando las mediciones. El paso final es obtener la estimación a posteriori de la covarianza del error.

Después de cada etapa de predicción y corrección, el proceso se repite con las estimaciones a posteriori previas siendo usadas para proyectar o estimar las nuevas estimaciones a priori. Esta naturaleza recursiva es una de las características más apreciadas del filtro de Kalman, no es necesario trabajar con todos los datos directamente en cada etapa como ocurre con el filtro de Weiner [16, 35].

4.2.2. Filtro de Kalman extendido

Como se describió anteriormente, el filtro de Kalman, aborda el problema de tratar de estimar el estado $x \in \Re^n$ de un proceso en tiempo discreto que es gobernado por una ecuación diferencial lineal estocástica. Cuando el proceso a estimar y/o, las relación entre las mediciones y el proceso son no lineales se recurre a una variación conocida como filtro de Kalman extendido.

El filtro de Kalman extendido linealiza el sistema en torno a una estimación de la media y la covarianza actual. En algo parecido a las series de Taylor, se puede linealizar la estimación en torno a la estimación actual usando las derivadas parciales de las funciones del proceso y de las mediciones.

Supóngase un proceso que tiene un vector de estados $x \in \Re^n$ controlado por una ecuación diferencial no lineal estocástica

$$x_k = f\left(x_{k-1}, u_{k-1}, w_{k-1}\right) \tag{4.16}$$

con una medición $z \in \Re^m$ que es

$$z_k = h\left(x_k, v_k\right) \tag{4.17}$$

donde las variables aleatorias w_k y v_k representan, otra vez, el ruido del proceso y el ruido de las mediciones respectivamente. La función no lineal f relaciona el estado x_{k-1} en el paso de tiempo anterior $k-1$ con el estado x_k en el paso actual k. Tiene como parámetros además una función controladora u_{k-1} y el ruido blanco del proceso w_{k-1}. La función no lineal h en la ecuación de medición (4.17) relaciona el estado x_k con la medición z_k.

En la práctica no se conocen los valores individuales del ruido w_k y v_k en cada paso. Sin embargo, se puede aproximar el vector de estados y el vector de mediciones sin el ruido como

$$\tilde{x_k} = f\left(\hat{x}_{k-1}, u_{k-1}, 0\right) \tag{4.18}$$

y

$$\tilde{z_k} = h\left(\tilde{x_k}, 0\right) \tag{4.19}$$

donde $\hat{x}_k$ es una estimación a posteriori del estado proveniente del paso anterior k.

Es importante mencionar que uno de los problemas fundamentales del filtro de Kalman extendido es que las distribuciones, o densidades en el caso continuo, de las diferentes variables aleatorias no son normales tras pasar por sus respectivas transformaciones no lineales.

Para estimar con diferencias y relaciones de mediciones no lineales, se comienza escribiendo nuevas ecuaciones de gobierno que linearizan una estimación sobre (4.18) y (4.19),

$$x_k \approx \tilde{x}_k + A\left(x_{k-1} - \hat{x}_{k-1}\right) + W w_{k-1} \tag{4.20}$$

$$z_k \approx \tilde{z}_k + H\left(x_k - \tilde{x}_k\right) + V v_k \tag{4.21}$$

donde

- x_ky z_k son los vectores de estado y medición actuales,
- $\tilde{x}_k$y $\tilde{z}_k$ son los vectores de estado y medición aproximados a partir de (4.18) y (5),
- $\hat{x}_k$ es la estimación a posteriori del estado en la etapa k,
- las variables aleatorias w_k y v_k representan el ruido del proceso y de la medición respectivamente,
- A es la matriz Jacobiana de derivadas parciales de f con respecto a x, o lo que es lo mismo

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}} \left(\hat{x}_{k-1}, u_{k-1}, 0\right), \tag{4.22}$$

- W es la matriz Jacobiana de derivadas parciales de f con respecto a w,

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}} \left(\hat{x}_{k-1}, u_{k-1}, 0\right), \tag{4.23}$$

- H es la matriz Jacobiana de derivadas parciales de h con respecto a x,

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}} \left(\tilde{x}_k, 0\right), \tag{4.24}$$

- V es la matriz Jacobiana de derivadas parciales de h con respecto a v,

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}} \left(\tilde{x}_k, 0\right). \tag{4.25}$$

Ahora definimos una nueva notación para el error estimado,

$$\tilde{e}_{x_k} = x_k - \tilde{x}_k, \tag{4.26}$$

y la innovación de la medición,

$$\tilde{e}_{z_k} = z_k - \tilde{z}_k. \tag{4.27}$$

En la práctica no se tiene acceso a x_k en (4.26), es el valor actual del vector estado. Por otra parte, si se tiene acceso a z_k en (4.27), es el valor actual de la mediciones que se usa para estimar x_k. Usando (4.26) y (4.27) se pueden escribir la ecuaciones de gobierno para un error de proceso como

$$\tilde{e}_{x_k} \approx A\left(x_{k-1} - \hat{x}_{k-1}\right) + \varepsilon_k, \tag{4.28}$$

$$\tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k, \tag{4.29}$$

donde ε_k y η_k representan nuevas variables aleatorias de media cero y covarianza WQW^T y VRV^T, con Q y R como en (4.3) y (4.4) respectivamente.

Cabe destacar que las ecuaciones (4.28) y (4.29) son lineales, y recuerdan a las ecuaciones (4.1) y (4.2) del filtro de Kalman discreto. Esto motiva el uso de el actual residuo de medición $\tilde{e}_{z_k}$ en (4.27) y un segundo (hipotético) filtro de Kalman para estimar el error de la predicción $\tilde{e}_{x_k}$ dado por (4.28). Esta estimación, denotada por $\hat{e}_k$, puede ser usada junto con (4.26) para obtener una estimación a posteriori del estado para el proceso no lineal original como

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k. \tag{4.30}$$

Las variables aleatorias de 4.28 y 4.29 tienen aproximadamente las siguientes distribuciones de probabilidad:

$$p\left(\tilde{e}_{x_k}\right) \sim N\left(0, E\left[\tilde{e}_{x_k}\tilde{e}_{x_k}^T\right]\right) \tag{4.31}$$

$$p\left(\varepsilon_k\right) \sim N\left(0, WQ_kW^T\right) \tag{4.32}$$

$$p\left(\eta_k\right) \sim N\left(0, VR_kV^T\right) \tag{4.33}$$

Dadas estas aproximaciones y dejando que el valor predicho de $\hat{e}_k$ sea cero, la ecuación del filtro de Kalman usado para estimar $\hat{e}_k$ es

$$\hat{e}_k = K_k\tilde{e}_{z_k}. \tag{4.34}$$

Sustituyendo (4.34) en (4.30) y haciendo uso de (4.27) se observa que realmente no es necesario el hipotético segundo filtro de Kalman:

$$\begin{aligned} \hat{x}_k &= \tilde{x}_k + K_k\tilde{e}_{z_k} \\ &= \tilde{x}_k + K_k\left(z_k - \tilde{z}_k\right) \end{aligned} \tag{4.35}$$

La ecuación (4.35) puede ser usada ahora para la predicción en el filtro de Kalman extendido, con $\tilde{x}_k$ y $\tilde{z}_k$ provenientes de (4.18) y (4.19), y una ganancia de Kalman K_k proveniente de (4.13) con la apropiada sustitución para la covarianza de error de medición.

El juego completo de ecuaciones del filtro de Kalman extendido se muestra en las tablas 4 y 5.

Como en el filtro de Kalman discreto, las ecuaciones de la etapa de predicción en la tabla 4 proyectan el estado y la covarianza estimados en la etapa anterior $k-1$ a la etapa actual k. Otra vez f viene de (4.18), A_k y W_k son las Jacobianas en la etapa k, y Q_k es la matriz de covarianza del proceso en la etapa k.

Al igual que en filtro de Kalman discreto, las ecuaciones de la etapa de corrección en la tabla 5 corrigen el estado y la covarianza estimadas con la medición z_k. Otra vez h

Tabla 4: ecuaciones de predicción del filtro de Kalman extendido

$$\hat{x}_k^- = f\left(\hat{x}_{k-1}, u_{k-1}, 0\right)$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$$

Tabla 5: ecuaciones de corrección del filtro de Kalman extendido

$$K_k^- = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T\right)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - h\left(\hat{x}_k^-, 0\right)\right)$$

$$P_k = \left(I - K_k H_k\right) P_k^-$$

proviene de (4.19), H_k y V son las Jacobianas de las mediciones en la etapa k, y R_k es la covarianza de la medición en la etapa k.

Una característica importante del filtro de Kalman extendido es que la Jacobiana H_k en la ecuación para la ganancia de Kalman K_k sirve para propagar o "magnificar" correctamente sólo la componente relevante de la información de la medida. Por ejemplo, si no existe una correspondencia directa entre la medición z_k y el estado mediante h, la Jacobiana H_k afecta a la ganancia de Kalman de forma que sólo magnifica la porción del residuo $z_k - h\left(\hat{x}_k^-, 0\right)$ que afecta al estado. Por supuesto si sobre todas las medidas no existe una correspondencia directa entre las mediciones z_k y el estado por medio de h, como cabría esperar, el filtro diverge rápidamente. En este caso el proceso no es observable [16, 35].

5. Estrategias de control

El desarrollo del control automático ha sido vital para el avance de la ingeniería y la ciencia. Además de su importancia en sistemas de vehículos espaciales, guiado de misiles, sistemas robóticos, y similares, el control automático se ha convertido en una parte importante de los procesos industriales y de manufacturación. Un ejemplo de ello son las máquinas herramienta de control numérico computerizado (CNC), el diseño de pilotos automáticos en la industrial aeroespacial, y en las cadenas de montaje de

la industria automovilística. También es esencial en las operaciones industriales como el control de presión, temperatura, humedad, viscosidad y caudal en las industrias de proceso.

El primer trabajo significativo en control automático fue el regulador de velocidad centrifugo de James Watt para el control de la velocidad de una máquina de vapor, en el siglo XVIII. Minorsky, Hazen y Nyquist, entre otros, realizaron importantes aportaciones en las etapas iniciales del desarrollo de la teoría de control. En 1922, Minorsky trabajó en los controladores automáticos para dirigir embarcaciones, y mostró que la estabilidad puede determinarse a partir de las ecuaciones diferenciales que describen el sistema. En 1932, Nyquist diseñó un procedimiento relativamente simple para determinar la estabilidad de sistemas en lazo cerrado, con base en la respuesta en lazo abierto en estado estable cuando la entrada aplicada es una senoidal. En 1934, Hazen, introdujo el término servomecanismos para los sistemas de control de posición, analizó el diseño de los servomecanismos con relevadores, capaces de seguir con precisión una entrada cambiante.

Durante la década de los cuarenta, los métodos de la respuesta en frecuencia hicieron posible que los ingenieros diseñaran sistemas de control lineales en lazo cerrado que cumplieran con unos requisitos de desempeño. A finales de los años cuarenta y principios de los cincuenta, se desarrolló por completo el método del lugar geométrico de las raíces propuesto por Evans.

Los métodos de respuesta en frecuencia y del lugar geométrico de las raíces, los cuales forman el núcleo de la teoría de control clásica, conducen a sistemas estables que satisfacen un conjunto más o menos arbitrario de requisitos de desempeño. En general, estos sistemas son aceptables pero no óptimos. Desde el final de la década de los cincuenta, el énfasis en los problemas de diseño de control se ha movido del diseño de sistemas que trabajen de manera apropiada al diseño de un sistema de control óptimo de algún modo significativo.

A medida que las plantas se vuelven más complejas, con múltiples entradas y salidas, la descripción del sistema de control requiere gran cantidad de ecuaciones. La teoría del control clásica, que trata de los sistemas con una entrada y una salida (Single Input-Single Output), pierde su solidez ante sistemas con entradas y salidas múltiples (Multiple Input-Multiple Output). Desde alrededor de 1960, debido al desarrollo de los computadores, se hizo posible el análisis en el dominio del tiempo de sistemas complejos, la teoría de control moderna, basada en el análisis en el dominio del tiempo y la síntesis a partir de variables de estados, se ha desarrollado para enfrentar la creciente complejidad de las plantas modernas y los requerimientos limitativos respecto de la precisión, el peso y el costo en aplicaciones militares, espaciales e industriales.

Entre la década de los 60 y los 80, se investigó a fondo el control óptimo tanto de sistemas determinísticos como estocásticos, y el control adaptativo, mediante el aprendizaje de sistemas complejos. Desde 1980 hasta la fecha, los descubrimientos en la teoría de control moderna se centraron en el control robusto, el control de H_∞, así como controladores adaptativos [21, 29].

En las siguientes secciones se resumen las principales características de las estrategias de control que han sido utilizadas en la literatura para el control de quadrotors.

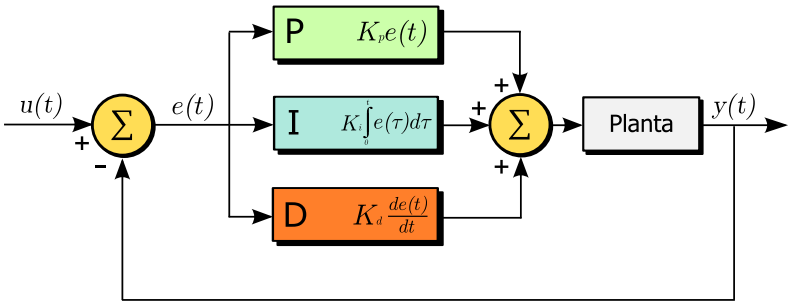

Figura 8: esquema de un controlador PID

5.1. Control Proporcional Integral Derivativo (PID)

Un PID (Proporcional Integral Derivativo) es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso. Es interesante señalar que más de la mitad de los controladores industriales que se utilizan hoy en día utilizan esquemas de control PID o PID modificados.

La utilidad de los controles PID estriba en que se aplican de forma casi general a la mayoría de los sistemas de control. En particular, cuando el modelo matemático de la planta no se conoce y, por tanto, no se pueden emplear métodos de diseño analíticos, es cuando los controles PID resultan más útiles. En el campo de los sistemas de control de procesos, es un hecho bien conocido que los esquemas de control básicos y modificados han demostrado su utilidad para aportar un control satisfactorio, aunque tal vez en muchas situaciones no aporten un control óptimo.

El control PID está compuesto, como se ve en la figura 8, de tres acciones que se suman entre sí: la acción de control proporcional, la integral y la derivativa. El valor proporcional determina la reacción al error actual. La acción integral genera una acción proporcional a la integral del error, esto nos asegura que aplicando un esfuerzo de control suficiente, de modo que el error de seguimiento se reduce a cero. La acción derivativa determina la reacción del tiempo en el que el error se produce.

La señal de salida estará determinada por el siguiente algoritmo

$$u\left(t\right)_{PID} = K_p e\left(t\right) + K_i \int_0^t e\left(\tau\right) d\tau + K_d \frac{de\left(t\right)}{dt}. \tag{5.1}$$

Ajustando las constantes de cada una de las acciones de control en el algoritmo de control del PID, el controlador puede proveer un control diseñado para lo que requiera el proceso a realizar. Como casi todos los controladores PID se ajustan en el sitio, en la literatura se han propuesto multitud de tipos diferentes de reglas de sintonización que permiten llevar a cabo una sintonización delicada y fina de los controladores PID

en el sitio. Asimismo, se han desarrollado métodos automáticos de sintonización y algunos PID poseen capacidad de sintonización automática en línea.

Un controlador PID puede ser llamado PI, PD, P o I en ausencia de las acciones de control respectivas. Los controladores PI son particularmente comunes, ya que la acción derivativa es muy sensible al ruido, y la ausencia de la acción integral puede evitar que se alcance el valor deseado.

Actualmente se usan en la industria formas modificadas del control PID, tales como I-PD y el control PID con dos grados de libertad. Es posible obtener muchos métodos prácticos para una conmutación sin choque (desde la operación manual hasta la operación automática) y una programación en aumento [29, 51].

Acción de control proporcional. La parte proporcional consiste en el producto entre la señal de error y la constante proporcional como para que hagan que el error en estado estacionario sea casi nulo, pero en la mayoría de los casos estos valores solo serán óptimos en una proporción determinada del rango total de control, siendo distintos los valores óptimos para cada porción del rango. Sin embargo, existe también un valor límite en la constante proporcional a partir del cual, en algunos casos, el sistema alcanza valores superiores a los deseados. Este fenómeno se llama sobreoscilación y, por razones de seguridad, no debe sobrepasar el 30 %, aunque es conveniente que la parte proporcional ni siquiera produzca sobreoscilación.

La fórmula de la acción proporcional esta dada por

$$u\left(t\right)_p = K_p e\left(t\right). \tag{5.2}$$

donde K_p es la ganancia proporcional.

El error, la banda proporcional y la posición inicial del elemento final de control se expresan en tanto por uno. Nos indicará la posición que pasará a ocupar el elemento final de control.

Hay una relación lineal continua entre el valor de la variable a controlada y la salida de la señal de control. La parte proporcional no considera el tiempo, por lo tanto, la mejor manera de solucionar el error permanente y hacer que el sistema contenga alguna componente que tenga en cuenta la variación respecto al tiempo, es incluyendo las acciones integral y derivativa [21, 29].

Acción de control integral. En el control proporcional de una planta, cuya función de transferencia no posee un integrador $\frac{1}{s}$, hay un error estacionario, o desplazamiento (*offset*), en la respuesta para una entrada escalón. Tal error se elimina si se incluye la acción integral del controlador.

En el control integral de una planta, la señal de control, que es la señal de salida a partir del controlador, es en todo momento el área bajo la curva de la señal de error hasta tal momento. La señal de control $u\left(t\right)$ tiene un valor diferente de cero cuando la señal de error $e\left(t\right)$ es cero como se aprecia en la figura 9 (a). Esto es imposible

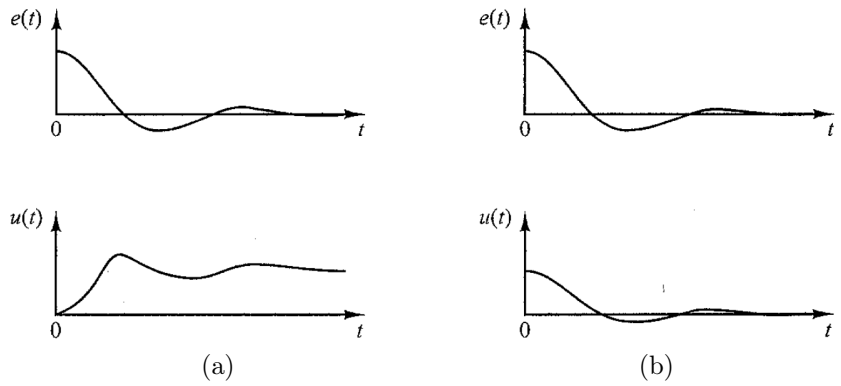

Figura 9: la gráfica (a) muestra una acción de control integral, mientras que la gráfica (b) muestra una acción de control proporcional

en el caso de un controlador proporcional, ya que una señal de control diferente de cero requiere una señal de error diferente de cero. Una señal de error diferente de cero en estado estacionario significa que hay una equivalencia. La figura 9 (b) muestra la curva $e\,(t)$ frente t y la curva $u\,(t)$ correspondiente frente t cuando el controlador es de tipo proporcional.

Obsérvese que la acción de control integral, aunque elimina el *offset* o el error en estado estacionario, puede conducir a una respuesta oscilatoria de amplitud decreciente lenta o, incluso, de amplitud creciente, y en ambos casos, por lo general, se consideran indeseables.

La fórmula de la acción integral esta dada por

$$u\,(t)_i = K_i \int_0^t e\,(t)\,dt. \tag{5.3}$$

El error es integrado, lo cual tiene como función promediarlo o sumarlo por un periodo determinado, para luego multiplicarlo por la constante K_i conocida como ganancia integral. Posteriormente, la respuesta integral es sumada a la respuesta proporcional para formar el control PI con el propósito de obtener una respuesta estable del sistema sin error estacionario [21, 29].

Acción de control derivativa. Cuando una acción de control derivativa se agrega a un controlador proporcional, aporta un modo de obtener un controlador con alta sensibilidad. Una ventaja de usar una acción de control derivativa es que responde a la velocidad de cambio del error y produce una corrección significativa antes de que la magnitud del error se vuelva demasiado grande. Por tanto, el control derivativo prevé el error, inicia una acción correctiva oportuna y tiende a aumentar la estabilidad del sistema.

Tabla 6: efectos de incrementar una constante de manera independiente

Constante	Tiempo de levantamiento	Sobrepaso	Tiempo de asentamiento	Error estado estable	Estabilidad
K_p	Disminuye	Aumenta	Cambio pequeño	Disminuye	Disminuye
K_i	Disminuye	Aumenta	Aumenta	Se elimina	Disminuye
K_d	Cambio pequeño	Disminuye	Disminuye	No afecta	Mejora si $K_d$es pequeña

La fórmula de la acción proporcional esta dada por

$$u\left(t\right)_d = K_d \frac{de\left(t\right)}{dt}. \tag{5.4}$$

Se deriva el error con respecto al tiempo y se multiplica por la ganancia derivativa K_d para luego ser sumado a las señales anteriores proporcional e integral.

Aunque el control derivativo no afecta de forma directa al error en estado estacionario, añade amortiguamiento al sistema y, por tanto, permite el uso de un valor más grande de la ganancia K_p, lo cual provoca una mejora en la precisión en estado estacionario.

Debido a que el control derivativo opera sobre la velocidad de cambio del error, y no sobre el error mismo, este modo nunca se utiliza solo. Siempre se emplea junto con una acción proporcional (PI) o proporcional-integral (PID). [21, 29]

5.1.1. Ajuste manual

Si el sistema se mantiene en línea y es posible modificar las constantes del controlador PID, se puede hacer un ajuste en línea modificando los parámetros con el fin de obtener el comportamiento deseado. Los efectos de incrementar cada una de las constantes independientemente se muestran en la tabla 6 [51].

5.1.2. Métodos de ajuste de Ziegler-Nichols

Ziegler y Nichols propusieron reglas para determinar las ganancias de un controlador PID: ganancias proporcional K_p, ganancia de integral K_i y ganancia derivativa K_d, basándose en las características de la respuesta transitoria de una planta dada. Hay dos métodos conocidos como reglas de ajuste de Ziegler-Nichols: el primer y el segundo método.

Primer método. En el primer método, la respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental. Si la planta no tiene integradores ni polos dominantes complejos conjugados, la curva de respuesta escalón unitario puede tener forma de S, como se observa en la figura 10. Este método se puede aplicar si la respuesta muestra una curva con forma de S. Tales curvas de respuesta

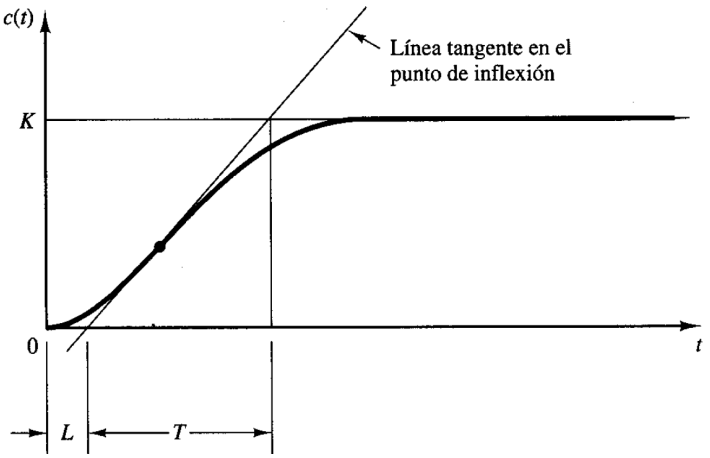

Figura 10: curva de respuesta en forma de S

Tabla 7: regla de sintonización de Ziegler-Nichols basada en la respuesta escalón de la planta (primer método)

Tipo de controlador	K_p	K_i	K_d
P	$\frac{T}{L}$	-	-
PI	$0,9\frac{T}{L}$	$\frac{0,3}{L}$	-
PID	$1,2\frac{T}{L}$	$\frac{1}{2L}$	$0,5L$

escalón se pueden generar experimentalmente o a partir de una simulación dinámica de la planta.

La curva con forma de S se caracteriza por dos parámetros: el tiempo de retardo L y la constante de tiempo T. El tiempo de retardo y la constante de tiempo se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de S y determinando las intersecciones de esta tangente con el eje del tiempo y con la línea $y(t) = K$, tal como se muestra en la figura 10. En este caso la función de transferencia $Y(s)/U(s)$ se aproxima mediante un sistema de primer orden con un retardo del modo siguiente:

$$\frac{Y(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \tag{5.5}$$

Ziegler y Nichols sugirieron establecer los valores de K_p, K_i y K_d de acuerdo con las fórmulas que se muestran en la tabla 7.

Obsérvese que el controlador PID sintonizado mediante el primer método de las reglas de Ziegler-Nichols produce

$$G_c(s) = K_p\left(1 + \frac{K_i}{s} + K_d s\right)$$

Tabla 8: regla de sintonización de Ziegler-Nichols basada en la ganancia crítica K_{cr} y en el periodo crítico P_{cr} (segundo método)

Tipo de controlador	K_p	K_i	K_d
P	$0,5K_{cr}$	-	-
PI	$0,45K_{cr}$	$1,2T_{cr}$	-
PID	$0,6K_{cr}$	$2T_{cr}$	$0,125T_{cr}$

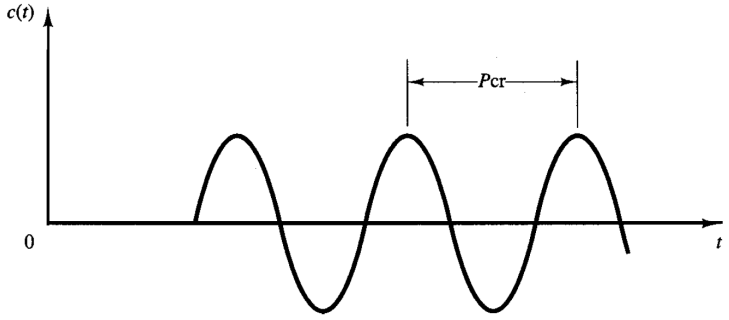

Figura 11: oscilación sostenida con un periodo P_{cr}

$$= 1,2\frac{T}{L}\left(1 + \frac{1}{2Ls} + 0,5Ls\right) \tag{5.6}$$

$$= 0,6T\frac{\left(s + \frac{1}{L}\right)^2}{s}$$

por tanto, el controlador PID tiene un polo en el origen y un cero doble en $s = {}^{-1}/_{L}$.

Segundo método. En el segundo método, primero se fija $K_i = 0$ y $K_d = 0$. Usando sólo la acción de control proporcional se incrementa K_p desde 0 hasta un valor crítico K_{cr}, a partir del cual la salida presente oscilaciones sostenidas. Si la salida no presenta oscilaciones sostenidas para cualquier valor de K_p entonces no se puede aplicar este método. Así, la ganancia crítica K_{cr} y el periodo P_{cr} correspondiente se determinan experimentalmente (figura 11). Ziegler-Nichols sugirieron que se establecieran los valores de los parámetros K_p, K_i y K_d de acuerdo con la fórmula que se muestra en la tabla 8.

Obsérvese que el controlador PID sintonizado mediante el segundo método de las reglas de Ziegler-Nichols produce

$$G_c\left(s\right) = K_p\left(1 + \frac{K_i}{s} + K_d s\right)$$

$$= 0,6K_{cr}\left(1+\frac{2}{P_{cr}s}+0,125P_{cr}s\right) \tag{5.7}$$

$$= 0,075K_{cr}P_{cr}\frac{\left(s+\frac{4}{P_{cr}}\right)^2}{s}$$

por tanto, el controlador, tiene un polo en el origen y un cero doble en $s = -4/P_{cr}$ [29].

5.1.3. Aplicación al control de quadrotors

En el artículo de Bouabdallah *et al.* [9] se utilizó un PID para control de la orientación de un micro helicóptero de cuatro rotores. El controlador PID obtuvo mejores resultados que el LQR pese a que el modelo matemático que se utilizó era más simple, lográndose el vuelo autónomo estable. El autor apunta a que esto es debido a la mayor tolerancia de este tipo de controladores a las variaciones de los parámetros del modelo.

Otro ejemplo de que el regulador PID es apropiado para el control de la posición y orientación de un quadrotor lo encontramos en Jun Li *et al.* [23], donde tras analizar la dinámica del quadrotor se diseña se implementa un regulador PID llegando a la conclusión de que la estabilización puede ser lograda mediante un regulador PID si los parámetros de este son los apropiados.

5.2. Control Lineal Cuadrático

El problema del control lineal cuadrático tiene su origen en el trabajo de Norbert Wiener sobre el filtrado cuadrático medio para el control de cañones antiaéreos durante la Segunda Guerra Mundial. Wiener empleó métodos basados en el dominio de la frecuencia para resolver este problema. Sin embargo, aportó como novedad importante un desarrollo teórico que permitía un método analítico para resolver el problema de diseño. Este planteamiento analítico contrastaba con los métodos de ensayos sucesivos con métodos gráficos basados en el criterio de estabilidad de Nyquist, que entonces se empleaba. El método Wiener permitía además tener en cuenta cuestiones tales como los ruidos de medida y otras perturbaciones de carácter aleatorio.

Las ideas de Wiener fueron reelaboradas durante los años 50 empleando la descripción interna de los sistemas y condujeron a lo que hoy se conoce como la teoría del control lineal cuadrático. De acuerdo con esta teoría el objetivo de un sistema de control es minimizar un índice de funcionamiento cuadrático. Se trata de mantener este sistema en un estado lo más cercano al reposo $x = 0$. El costo correspondiente a las desviaciones del estado de reposo se expresa por

$$J_1 = \int_0^T x^T Q x dt + x^T(T) S x(T) \tag{5.8}$$

sujeto a las restricciones que representan un sistema lineal

$$\dot{x} = Ax + Bu. \tag{5.9}$$

Lo que recibe la denominación de problema del regulador lineal cuadrático o problema LQR (acrónimo de Linear Quadratic Regulator). Su solución se reduce a la de una ecuación diferencial de Ricatti. Durante el periodo de 1960-70 se desarrollaron muchos estudios teóricos sobre este problema. Las ventajas que presenta la solución de este problema sobre las técnicas de diseño clásicas son las siguientes:

- permite la optimización para intervalos de tiempo finito, los métodos en el dominio de la frecuencia de Wiener estaban limitados a intervalos de optimización infinitos,
- son aplicables a sistemas que varían con el tiempo, los métodos en el dominio de la frecuencia están limitados a sistemas invariantes en el tiempo,
- permiten abordar de forma relativamente simple el problema de los sistemas multivariable.

Sin embargo, la teoría del LQR no aborda dos cuestiones muy importantes que aparecen en el diseño de sistemas de control realimentados: la falta de precisión en el modelo de la planta y el ruido en los sensores. Además, la teoría de LQR presupone el conocimiento del estado del sistema que es frecuente que no esté disponible. El problema lineal cuadrático con perturbaciones aleatorias se reduce a la resolución de dos ecuaciones de Ricatti desacopladas, ya que se puede demostrar que es posible separar el problema en dos: el problema de control óptimo con realimentación del estado, tal y como se aborda en la teoría LQR, y el problema de la estimación del estado. Esta separación se puede justificar teóricamente en el caso de que las perturbaciones estocásticas sean gausianas, por lo que el problema lineal cuadrático estocástico se conoce comúnmente como el problema lineal cuadrático gausiano (LQG) [3].

Como se comentó anteriormente, en Bouabdallah *et al.* [9], se presentan los resultados de aplicar dos técnicas de estrategias, PID y LQR, al control de la orientación de un micro helicóptero de cuatro rotores. Para el desarrollo del controlador LQR se utilizó un modelo dinámico más complejo que el que se utilizó para el diseño del PID. Sin embargo no se logró un vuelo libre estable con el controlador LQR. Este hecho se achaca a que no se tuvo en cuenta la dinámica de los actuadores, siendo el controlador LQR más sensible a los errores de modelado.

5.3. Control difuso

La lógica difusa ha surgido como una herramienta para el control de subsistemas y procesos industriales complejos, así como también para la electrónica de entretenimiento y hogar, sistemas de diagnóstico y otros sistemas expertos. Aunque la lógica borrosa se inventó en Estados Unidos, el rápido crecimiento de esta tecnología ha

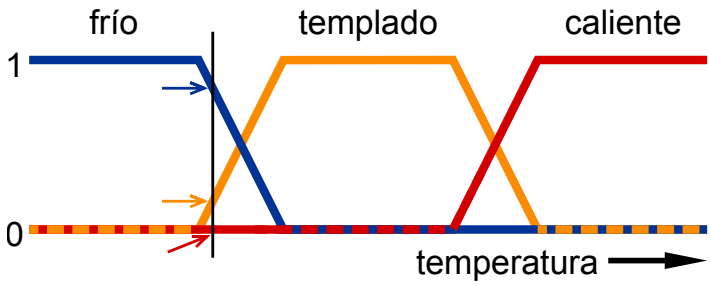

Figura 12: escala difusa de temperatura

comenzado en Japón y ha alcanzado nuevamente a los Estados Unidos, y también Europa.

La lógica borrosa es básicamente una lógica multievaluada que permite valores intermedios para poder definir evaluaciones convencionales como sí/no, verdadero/falso, blanco/negro, etc. Las nociones como "más bien frío" o "poco caliente" pueden formularse matemáticamente y ser procesadas por computadoras. De esta forma se ha realizado un intento de aplicar una forma de pensar más humana en la programación de computadoras. La lógica borrosa se inició en 1965 por Lotti A. Zadeh, profesor de la Universidad de California en Berkley.

La lógica difusa y la lógica probabilística son matemáticamente similares – ambos tienen valores verdaderos entre 0 y 1– pero conceptualmente distintos. La lógica difusa corresponde a "grados de verdad", mientras que la lógica probabilística corresponde a "probabilidad"; ya que son diferentes, la lógica difusa y la lógica probabilística pueden producir diferentes modelos de las mismas situaciones del mundo real.

Tanto los grados de verdad como las probabilidades oscilan entre 0 y 1 y por lo tanto pueden parecer similares al principio. Por ejemplo, supóngase que una botella de 100 ml contienen 30 ml de agua. Entonces se pueden considerar dos conceptos: vacío y lleno. El significado de cada uno de ellos puede ser representado por un conjunto difuso determinado. Entonces se podría definir la botella como 0,7 de vacío y 0,3 de lleno. Hay que tener en cuenta que el concepto de vacío sería subjetivo y por lo tanto dependerá del observador o diseñador. Es esencial darse cuenta de que la lógica difusa utiliza grados de verdad como un modelo matemático del fenómeno de la vaguedad, mientras que la probabilidad es un modelo matemático de la ignorancia.

Una aplicación básica debe caracterizar subrangos de una variable continua. Por ejemplo, la temperatura medida para el antibloqueo de un freno puede tener varias funciones de pertenencia separadas que definen determinados intervalos de temperatura necesarios para controlar los frenos correctamente. Cada función asigna el valor de temperatura igual a un valor de verdad en el intervalo de 0 a 1. Estos valores de verdad se puede utilizar entonces para determinar cómo deben ser controlados los frenos.

En la figura 12, el significado de las expresiones frío, templado y caliente están representadas por las funciones que aplican una escala de temperatura. Un punto en

esa escala tiene tres valores de verdad, uno para cada una de las tres funciones. La línea vertical de la figura representa una temperatura particular. La flecha roja en cero, esta temperatura puede ser interpretada como "no caliente". La flecha de color naranja (señalando a 0,2) puede describirla como "un poco caliente" y la flecha azul (señalando a 0,8) "bastante frío".

Mientras que las variables en matemáticas suelen tener valores numéricos, en lógica difusa, las variables lingüísticas no numéricas se utilizan a menudo para facilitar la expresión de reglas y hechos. Una variable lingüística, como la edad, pueden tener un valor como el joven o su antónimo viejo. Sin embargo, la gran utilidad de variables lingüísticas es que pueden ser modificados mediante los modificadores lingüísticos aplicados a términos primarios. Los modificadores lingüísticos puede estar asociado con ciertas funciones.

La teoría de conjuntos difusos define operadores difusos en los conjuntos difusos. El problema en la aplicación de esto es que el operador difuso apropiado puede no ser conocido. Por esta razón, la lógica difusa por lo general utiliza reglas SI-ENTONCES, o construcciones que sean equivalentes, tales como matrices asociativas difusas. Las reglas se expresan normalmente de la manera: SI variable ES propiedad ENTONCES acción. Por ejemplo un regulador de temperatura que usará un ventilador podría quedar así:

```
SI temperatura ES muy fría ENTONCES parar ventilador
SI temperatura ES fría ENTONCES reducir velocidad ventilador
SI temperatura ES normal ENTONCES mantener velocidad ventilador
SI temperatura ES caliente ENTONCES aumentar velocidad ventilador
```

No hay un DE LO CONTRARIO, todas las reglas son evaluadas puede haber un "frío" y "normal" a la vez con diferentes grados [44].

La lógica difusa se ha aplicado al control del los quadrotors, prueba de ello es el artículo de Z. Lendek *et al.* [22], donde se diseña un estimador no lineal Takagi-Sugeno (TS) para la estimación de la orientación y velocidad angular de un quadrotor. El resultado es que el estimador no lineal TS, el cual usa la lógica difusa para hacer frente a las no linealidades del sistema a observar, ofrece mejores resultados que el estimador lineal este último se aleja del punto de equilibrio.

5.4. Control robusto

El control robusto es una rama de la teoría de control que explícitamente se refiere a la incertidumbre en su enfoque en el diseño del controlador. Los controladores robustos están diseñados para funcionar adecuadamente siempre que los parámetros inciertos o las alteraciones se encuentren dentro de un conjunto, generalmente compacto. Los métodos de control robusto tratan de conseguir resultados sólidos y/o la estabilidad en presencia de errores de modelado acotados.

Los primeros métodos de Bode y otros eran bastante robustos; la representación en espacio de estados carecía en ocasiones de una falta de robustez que llevo a la investigación para su mejora. Este fue el comienzo de la teoría de control robusto que tomo forma en la década de los 80 y a día de hoy sigue activa.

En contraste con la política de control adaptativa, la política de control robusta es estática. En vez de adaptarse a las variaciones de las mediciones, el controlador está diseñado para trabajar suponiendo que ciertas variables serán desconocidas pero acotadas.

De manera informal, se dice que un controlador diseñado para un conjunto particular de parámetros es robusto si también trabaja bien bajo un conjunto diferente de suposiciones. La realimentación con alta ganancia es un ejemplo simple de método de control robusto. Con una ganancia lo suficientemente alta, los efectos de la variación de cualquier parámetro serán despreciables. La realimentación con alta ganancia es el principio que permite usar modelos simplificados de amplificador operacional y transistores bipolares de emisor degenerado en gran variedad de situaciones.

La teoría de control robusto comenzó a finales de los 70 y pronto desarrollo numerosas técnicas para lidiar con sistemas con incertidumbres acotadas. Probablemente la técnica de control robusto más importante es la H_∞, la cual fue desarrollada por Duncan McFarlane y Keith Gobler de la Universidad de Cambridge. Este método minimiza la sensibilidad del sistema sobre su espectro de frecuencias, y garantiza que el sistema no se desviará mucho de la trayectoria cuando las alteraciones entren en el sistema.

Un área emergente en el control robusto desde el punto de vista de su aplicación es control por modos deslizantes (SMC), que es una variación de la estructura de control variable (VSS). La robustez con la que el control por modos deslizantes maneja la incertidumbre, así como la simplicidad en el diseño atrajo una variedad de aplicaciones. [53]

En el artículo de G. V. Raffo *et al.* [31] se presenta una estrategia de control no lineal robusta para el problema del seguimiento de trayectorias de un vehículo aéreo no tripulado del tipo quadrotor. El controlador H_∞ es utilizado para estabilizar los movimientos rotacionales. La robustez del controlador H_∞ se pudo observar cuando se consideró la incertidumbre en los términos inerciales. Esta robustez fue corroborada en las simulaciones.

5.5. Control predictivo

El control predictivo tiene como objetivo resolver de forma efectiva problemas de control y automatización de procesos industriales que se caracterizan por presentar un comportamiento dinámico complicado, multivariable, y/o inestable. La estrategia de control en la que se basa este tipo de control utiliza el modelo matemático del proceso a controlar para predecir el comportamiento futuro de dicho sistema, y en base a este comportamiento futuro puede predecir la señal de control futura.

El control predictivo integra disciplinas como el control óptimo, control estocástico, control de procesos con retardo de tiempo, control multivariable o control con restricciones.

Existen multitud de algoritmos de control predictivo que han sido aplicados con éxito. Uno de ellos es el Control Predictivo Basado en Modelo (CPBM), también conocido como Model Predictive Control (MPC). El control predictivo basado en modelo se puede definir como una estrategia de control que se basa en la utilización explícita de un modelo matemático interno del proceso a controlar (modelo de predicción), el cual se utiliza para predecir la evolución de las variables a controlar a lo largo del horizonte temporal de predicción especificado por el operador. De este modo se pueden calcular las variables manipuladas futuras (señal de control futura) para lograr que en dicho horizonte las variables controlada converjan a sus respectivos valores de referencia.

El MPC se enmarca dentro de los controladores óptimos, es decir, aquellos en los que las actuaciones responden a la optimización de un criterio. El criterio a optimizar, o función de coste, está relacionado con el comportamiento futuro del sistema que se predice gracias al denominado modelo de predicción.

El intervalo de tiempo futuro que se considera en la optimización se denomina horizonte de predicción. Dado que el comportamiento futuro del sistema depende de las actuaciones que se aplican a lo largo del horizonte de predicción, son estas variables de decisión respecto a las que se optimiza el sistema. La aplicación de estas actuaciones sobre el sistema conducen a un control en bucle abierto.

La posible discrepancia entre el comportamiento predictivo y el comportamiento real del sistema crean la necesidad de imponer cierta robustez al sistema incorporando una realimentación al mismo. Esta realimentación se consigue gracias a la técnica de horizonte deslizante que consiste en aplicar las actuaciones obtenidas durante un periodo de tiempo, tras el cual se muestra el estado del sistema y se resuelve un nuevo problema de optimización. De esta manera el horizonte de predicción se va deslizando a lo largo del tiempo.

Una de las propiedades más atractivas del MPC es su formulación abierta, que permite la incorporación de distintos tipos de modelos de predicción, sean lineales o no lineales, monovariables o multivariables, y la consideración de restricciones sobre las señales del sistema. Esto hace que sea una estrategia muy utilizada en diversas áreas de control. El MPC es una de las pocas técnicas de control que permiten controlar sistemas con restricciones incorporando éstas en el propio diseño del controlador.

Estas características han hecho del control predictivo una de las escasas estrategias de control avanzado con un impacto importante en problemas de ámbito industrial. Por tal motivo es importante resaltar que el control predictivo se ha desarrollado en el mundo de la industria, y ha sido la comunidad investigadora la que se ha esforzado en dar un soporte teórico a los resultados prácticos obtenidos [34, 48].

En el artículo de K. Alexis *et al.* [2] se diseña una estrategia de control predictivo basada en modelo para el control de la orientación y la posición del quadrotor. El sistema desarrollado es capaz de rendir de manera precisa en vuelos en interior mediante la utilización de un sistema de fusión de sensores IMU/Sonar/IMU. La novedad en de este artículo residía en la propuesta del control MPC, siendo el resultado experimental satisfactorio.

5.6. Control por modos deslizantes

En teoría de control, los modos deslizantes son una forma de control de estructura variable. Es un método de control no lineal que altera la dinámica de un sistema no lineal aplicando un control alternante. La ley de control no es una función constante en el tiempo. En cambio, va alternando de una estructura continua a otra basándose en la posición en ese instante en el espacio de estados. Las estructuras de control múltiple se diseñan para que las trayectorias siempre se muevan hacia la condición de alternancia y, así, la trayectoria última no existirá completamente en una sola estructura de control. Lo que ocurrirá es que la trayectoria última se deslizará por la frontera entre las diferentes estructuras de control.

La mayor ventaja del control por modo deslizante es su robustez, ya que el control puede ser tan simple como el salto entre dos estados, no necesita ser preciso y no se verá afectado a variaciones de parámetros que entren en el canal de control. Además, como la ley de control no es una función continua, el modo deslizante se puede alcanzar en tiempo finito.

El control en modo deslizante debe ser aplicado con mucho más cuidado que otras formas de control no lineal que poseen una acción de control más moderada. En particular, porque los actuadores tienen retardos y otras imperfecciones, y el modo de control puede llegar a chattering, perdida de energía, daño en la planta y excitación de dinámicas no modeladas [7, 27].

El control por modos deslizantes se utilizo para la estabilización de un quadrotor en presencia de fallos en los actuadores en el artículo de H. Khebbache *et al.* [20]. Los resultados de la simulación demostraron una gran eficiencia del la estrategia de control al mantener la estabilidad del quadrotor a pesar del fallo en los actuadores.

Parte III
Materiales y métodos

6. Materiales

A lo largo de este capítulo se describirán brevemente los distintos materiales que han sido necesarios para la realización de este proyecto. Se comenzará con del Draganflyer SAVS, que es quadrotor utilizado como plataforma, para continuar con el microcontrolador y los sensores utilizados para poder estimar la orientación del quadrotor. Por último se analizará del software utilizado en la simulación.

6.1. Draganflyer SAVS

El Dranganflyer SAVS (Stabilized Aerial Video System) es un pequeño quadrotor radiocontrol diseñado para obtener imágenes desde el aire gracias a la cámara a color que incorpora (ver figura 13). Fue lanzado al mercado en el año 2006 por la empresa Draganfly Inovantions Inc. que está dedicada a la fabricación de modelos radiocontrol aéreos. El Dranganflyer SAVS no se comercializa actualmente al haber sido sustituido por el Dragonflyer X4.

Su tamaño es reducido, posee un diámetro de 75cm lo que posibilita su vuelo en interiores. La estructura del Dragonflyer SAVS está constituida de fibra de carbono y nailon de alto-impacto gracias a los cuales su peso es de sólo 540g. Para poder volar utiliza cuatro motores con escobillas de alta velocidad y una transmisión de dos engranajes.

Para lograr un vuelo estable el quadrotor utiliza tres giróscopos piezoeléctricos y el sistema patentado Thermal Intelligence (TI). Cada uno de los tres giroscopios mide la velocidad angular en uno de los tres ejes (alabeo, cabeceo y guiñada). El sistema Thermal Intelligence funciona utilizando cuatro sensores infrarrojos que miden la diferencia de temperatura entre el cielo y la tierra. Usando como entrada estos sensores la CPU embarcada automáticamente hace que el quadrotor se nivele cuando se suelta el control.

Las imágenes aéreas son grabadas por una cámara digital de 480 lineas y transmitidas a las estación base con un transmisor de 50mW de potencia que funciona a 2,4Ghz. La estación base se encuentra en un maletín de transporte y para recibir la señal utiliza dos antenas “patch” con una ganancia de 8dbi y nos proporciona salidas RCA para vídeo y audio [41].

6.2. Arduino

Arduino es plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo que tienen como objetivo facilitar el uso de la electrónica en proyectos multidisciplinares.

Figura 13: Draganflyer SAVS

El hardware consiste en una placa Arduino con un procesador Atmel AVR y puertos de entrada y de salida. El software consiste en un lenguaje estándar de programación y un cargador de arranque (boot loader) que corre en la placa. El hardware Arduino se programa usando un lenguaje basado en Wiring (sintaxis y librerías), similar a C++ con algunas ligeras simplificaciones y modificaciones, y un entorno de desarrollo integrado basado en Processing (ver figura 15) [40].

Las versiones actuales pueden comprarse ensambladas; el diseño de hardware está disponible en caso de que se desee montar un Arduino a mano. Además, terceras empresas han lanzado variaciones de los Arduinos hechos en Italia, con diferentes niveles de compatibilidad, y algunos de ellos se programan usando el software Arduino.

Una placa Arduino consiste en un microcontrolador Atmel AVR de 8-bit con componentes complementarios que facilitan su programación e incorporación en otros circuitos. Un aspecto importante de Arduino es la manera en el que los conectores están distribuidos en la placa, permitiendo conectar una gran variedad una gran variedad de añadidos, conocidos como *shields*, a la placa principal. Algunas de estas *shields* se comunican directamente con la placa Arduino usando varios terminales, pero muchas de ellas se comunican individualmente a través del bus I^2C, permitiendo conectar varias *shields* simultáneamente y usarlas en paralelo.

Los Arduino oficiales usan la serie de controladores megaAVR, específicamente el ATmega8, ATmega168, ATmega328, ATmega1280, y el ATmega2560. Una gran variedad de procesadores se han usado en los Arduinos compatibles, un ejemplo de ello es el

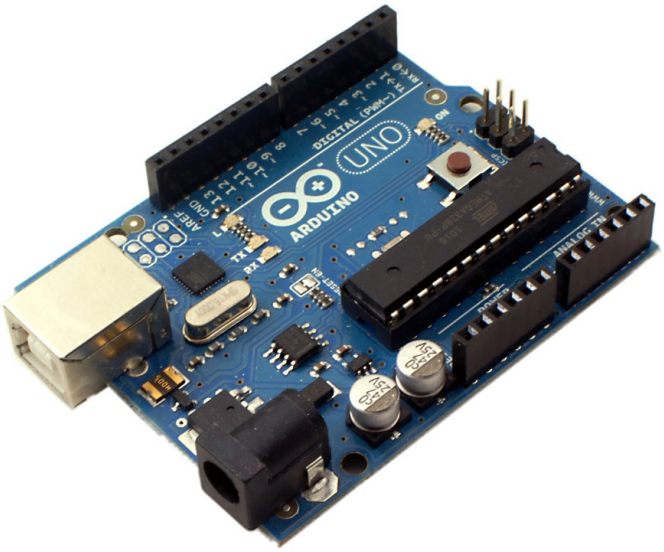

Figura 14: placa Arduino UNO

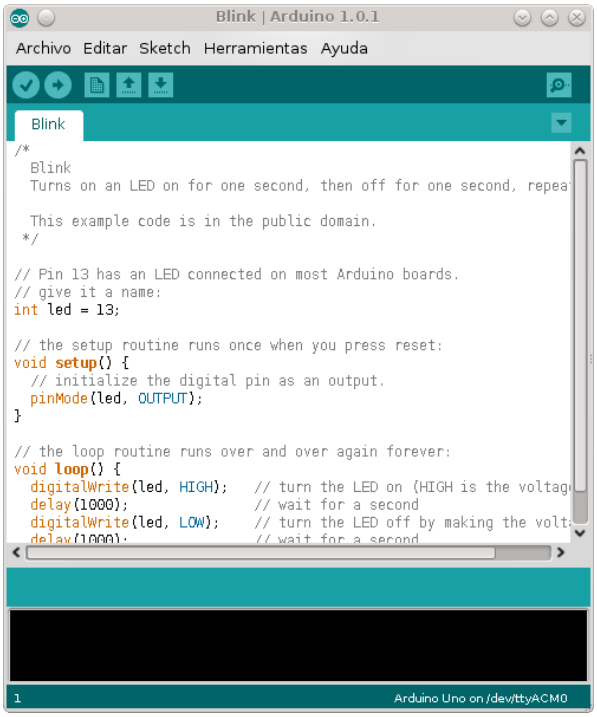

Figura 15: entorno de desarrollo integrado Arduino

Mapple de LeafLabs el cual usa un microcontrolador ARM Cortex M3. La mayoría de las placas incluyen un regulador de 5 voltios y un cristal oscilador de 16 MHz, o, en algunas variaciones, resonador cerámico. Un microcontrolador Arduino es preprogramado con un cargador de arranque que simplifica el proceso de cargar los programas en la memoria del controlador, comparado con otros dispositivos que normalmente necesitan un programador externo.

A nivel conceptual, cuando se usa el software proporcionado por Arduino, todas las placas son programadas mediante una conexión de serie RS-232, pero la forma en la que está implementado varía en función de la versión del hardware. Las placas Arduino con puerto de serie poseen un circuito inversor simple que convierte señales RS-232 y TTL. Las placas Arduino actuales son programadas a través de USB, implementado usando un adaptador USB a serie como el FTDI FT232 [39].

En este proyecto se ha utilizado una placa Arduino UNO (ver figura 14) que tiene las siguientes características:

- microcontrolador ATmega328P de 8-bit funcionando a 16MHZ,
- memoria flash de 32 kB,
- memoria EEPROM de 1 kB,
- memoria SRAM de 2 kB,
- 14 terminales de entrada/salida digitales, 6 de ellos con capacidad de producir señales PWM,
- 6 terminales analógicos,
- interfaz USB mediante microcontrolador ATmega 16U2.

6.3. Sensores

La estimación del estado (actitud, velocidad y posición) es una necesidad fundamental para los vehículos autónomos. La precisión que se demanda en la estimación del estado depende en cualquier caso del tipo de vehículo, del sistema de control usado y de la aplicación.

Para la navegación se utilizan una gran variedad de sensores. Algunos se remontan a tiempos remotos como es el caso de la brújula. En general, existe una gran cantidad de sensores (presión, velocidad del aire, temperatura, etc.), siendo los más importantes para la estimación del estado los sensores inerciales: acelerómetros, giroscopios y magnetómetros (brújula).

Los sensores inerciales MEMS (siglas en inglés de *Micro-machines ElectroMechanical Systems*) son fruto del desarrollo de los avances en el campo de los semiconductores dando lugar a circuitos integrados con características tridimensionales e incluso con piezas móviles. Esto ha permitido el desarrollo de sensores con una disminución de coste, tamaño y consumo enorme respecto a los sensores tradicionales.

6.3.1. Acelerómetro

Un acelerómetro es un instrumento que mide la aceleración. La aceleración medida está relacionada con el peso de una masa de prueba. Por ejemplo, un acelerómetro situado en un cohete lejos de cualquier campo gravitacional (se supone gravedad cero) que está acelerando debido al empuje del motor, medirá el ritmo de cambio de la velocidad del cohete relativa a cualquier marco de referencia inercial. Si el acelerómetro se encontrase en la superficie terrestre mediría una aceleración de valor g con dirección vertical cuando se encontrase estacionario, debido a que las masas en la Tierra tienen un peso mg. Sin embargo en este caso no hay un cambio en la velocidad.

De esta forma se podría modelar la medida del acelerómetro f como

$$\vec{f} = \vec{a} + \vec{g} = \frac{\vec{F}}{m} + \vec{g}, \tag{6.1}$$

por tanto, la medida en las siguientes situaciones será

- objeto estacionario sobre la superficie terrestre: $\vec{a} = 0$ y $\vec{f} = \vec{g}$.
- objeto en caída libre: $\vec{a} = -\vec{g}$ y $\vec{f} = 0$.

Conceptualmente, un acelerómetro se comporta como una masa amortiguada con un muelle. Cuando el acelerómetro experimenta una aceleración, la masa se desplaza hasta el punto en el que el muelle es capaz de acelerar la masa al mismo ritmo que la carcasa. El desplazamiento es medido para obtener la aceleración.

En dispositivos comerciales, para convertir el movimiento mecánico en una señal eléctrica se utilizan normalmente componentes piezoeléctricos, piezoresistivos y capacitativos. Los acelerómetros piezoeléctricos se basan en piezocerámicos o en cristales como el cuarzo. Son insuperables a altas frecuencias, bajo peso del dispositivo y amplio rango de temperatura. Los acelerómetros piezoresistivos se prefieren para aplicaciones con grandes impactos. Los acelerómetros capacitativos usan normalmente un sensor de silicio micro mecanizado. Su rendimiento es superior a bajas frecuencias y pueden lograr una gran estabilidad y linealidad.

Los acelerómetros modernos son a menudo pequeños sistemas microelectromecánicos (MEMS), y de hecho son los dispositivos MEMS más simples (ver figura 16), que constan con poco mas que una viga en voladizo con una masa de prueba (también conocida como masa sísmica). El amortiguamiento es el resultados del gas sellado dentro del dispositivo.

Bajo la influencia de aceleraciones externas la masa de prueba se desvía de su posición neutra. Esta desviación es medida de manera analógica o digital. Comúnmente se mide la capacitancia entre un conjunto de vigas fijas y un conjunto de vigas con una masa de prueba. Este método es simple, seguro y barato. Integrar piezoresistencias en el resorte para detectar la deformación de este, y en consecuencia el desplazamiento, es una buena alternativa, sin embargo son necesarios más pasos durante el proceso de

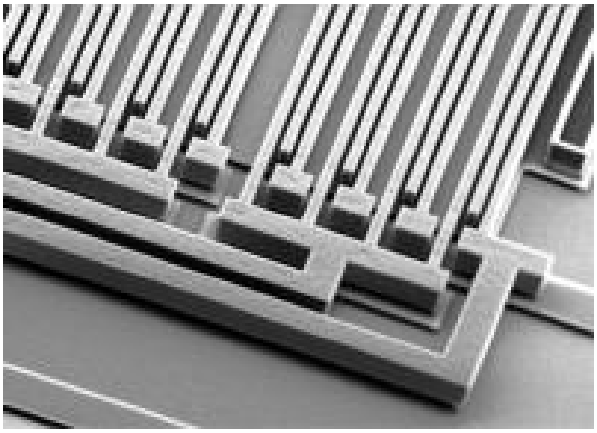

Figura 16: estructura de un acelerómetro MEMS

fabricación. Para grandes sensibilidades se utiliza el efecto túnel; requieren un proceso de fabricación delicado y muy caro. La medición óptica se ha demostrado a nivel de laboratorio.

La mayoría de los acelerómetros micromecánicos operan en plano, es decir, están diseñados para ser sensible solo en una dirección contenida en el plano del *die*. Integrando dos dispositivos perpendicularmente en un único *die* se fabrican los acelerómetros biaxiales. Añadiendo otro dispositivo fuera del plano se obtienen los acelerómetros triaxiales. Estas combinaciones presentan un error de alineamiento mucho menor que tres dispositivos individuales combinados después del encapsulado [8].

Para medir la aceleración en el quadrotor se ha utilizado un acelerómetro triaxial ADXL345 de la compañía Analog Devices (ver figura 17). Este acelerómetro forma parte de la IMU Fusion Board de Sparkfun. Sus principales características son:

- rango de medida de ± 2, ± 4, ± 8 y ± 16 g,
- resolución de 10-bits,
- consumo de 40 μA en modo medida y 0,1 μA en modo standby.
- detector de caída libre,
- interfaz SPI e I^2C.

6.3.2. Giroscopio

Un giroscopio es un instrumento, basado en los principios del momento angular, para medir o mantener la orientación. Mecánicamente, un giroscopio es una rueda o disco giratorio en el que el eje es libre de adoptar cualquier orientación. Aunque esta orientación no permanece fija, cambia en respuesta a un par externo mucho menos y en

Figura 17: acelerómetro Analog Devices ADXL345

una dirección diferente de lo que lo haría sin el momento angular asociado al disco con gran velocidad de giro y momento de inercia. Como el par externo es minimizado por el montaje del dispositivo en cardanes, su orientación permanece casi fija, con independencia de cualquier movimiento de la plataforma en la que se monta.

Un giroscopio exhibe un número de comportamiento incluyendo la precesión y la nutación. Los giroscopios pueden ser usados para construir girocompases, los cuales complementa o remplazan a los compases magnéticos, para asistir en la estabilidad o como parte de un sistema de medición inercial. Los efectos giroscópicos se utilizan en boomerangs y yoyos.

La ecuación fundamental que describe el funcionamiento de un giroscopio es

$$\vec{\tau} = \frac{d\vec{L}}{dt} = \frac{d\left(I\vec{\omega}\right)}{dt} = I\vec{\alpha} \tag{6.2}$$

donde los pseudovectores $\vec{\tau}$ y $\vec{L}$ son, respectivamente, el par en el giroscopio y su momento angular, el escalar I es su momento de inercia, el vector ω es su velocidad angular, y el vector $\vec{\alpha}$ su aceleración angular.

Los giroscopios con tecnología MEMS toman ventaja del efecto Coriolis. En un marco de referencia rotando a una velocidad angular Ω, una masa M moviéndose con velocidad v se ve sometida a una fuerza

$$F = 2M\vec{v} \times \vec{\Omega}. \tag{6.3}$$

En la literatura aparecen gran cantidad de tipos de giroscopios MEMS, aunque la mayoría caen en las categorías de giroscopio de tenedor ajustado (tuning-fork), volante oscilante, péndulo de Foucault, y resonador Helmholtz. Los giroscopios convencionales de rueda giratoria, no MEMS, son comunes, pero la levitación y rotación en dispositivos MEMS sin muelles no ha sido comercializado aún.

Los giroscopios de tenedor ajustado contienen un par de masas que están forzadas a oscilar con igual amplitud pero en direcciones opuestas. Cuando rotan, la fuerza de Coriolis crea una vibración ortogonal que puede ser sentida de diferentes maneras.

La rotación provoca que las masas de prueba vibren fuera del plano, siendo este movimiento es detectado por sensores capacitativos. Esta tecnología ha sido patentada por Rockwell, Boeing, Honeywell y otros.

Los modos resonantes de un sensor MEMS inercial son extremadamente importantes. En un giroscopio, normalmente hay un modo de vibración que es provocado y un segundo modo como salida detectada. En algunos casos, los modos de entrada y salida son degenerados o están cerca de serlo. Si los modos de entrada y salida son elegidos de manera que estos se encuentren separados por un 10 % aproximadamente, la sensibilidad en bucle abierto se incrementará debido a efectos de resonancia. Es crítico que no haya otros modos de resonancia cercanos a las frecuencias de resonancia de entrada y salida.

Figura 18: IMU Fusion Board de Sparkfun

En los giroscopios de disco oscilante, el disco es forzado a oscilar sobre su eje de simetría, una rotación en cualquier otro plano provoca una inclinación en el disco. Este cambio es detectado con electrodos capacitativos situados debajo del disco. Es posible medir dos ejes de rotación con un solo disco oscilante.

Un tercer tipo de giroscopio son los de resonador Helmholtz. Fabricados de silicio fundido, estos dispositivos son también conocidos como giroscopios de resonancia hemisférica. En un giroscopio de resonador de Helmholtz, el anillo resonante es conducido a la resonancia y la posición de los puntos nodales indican la velocidad de rotación. Los modos de entrada y salida son normalmente degenerados, pero debido a imperfecciones en el mecanizado son requeridos algunos ajustes.

Los giroscopios de péndulo de Foucault son dispositivos que se basan en una varilla vibratoria que está orientado típicamente fuera del plano del chip. Son por lo tanto, un reto de construir con las herramientas de fabricación normales, pero los recientes avances en la tecnología MEMS permite la relación de aspecto muy alto que hacen que sea posible fabricar el péndulo, sin montaje manual de la varilla [8].

En el quadrotor usamos como giroscopio la IMU-3000 de InvenSense que se encuentra integrado en la IMU Fusion Board de Sparkfun (ver figura 18). Este giroscopio triaxial digital tiene las siguientes características:

- rango programable de $\pm 250^{\circ}/s$, $\pm 500^{\circ}/s$, $\pm 1000^{\circ}/s$ y $\pm 2000^{\circ}/s$,
- trabaja con conversores analógico a digital de 16-bits,

- consumo de 6,1 mA con los tres ejes activos y de 5 μA en modo dormido,
- filtro paso bajo implementado en el sensor,
- interfaz I^2C.

6.3.3. Magnetómetro

Los magnetómetros permiten determinar la fuerza o la dirección de un campo magnético. Los magnetómetros se pueden clasificar en dos tipos básicos: escalares, miden la fuerza del campo magnético al que se encuentran sometidos, y vectoriales, miden la componente de campo magnético en una dirección concreta. Estos últimos son conocidos como brújulas o compás y son los más interesantes para la navegación. Entre las tecnologías de magnetómetros destacar los siguientes tipos:

- Bobina rotatoria. El campo magnético induce una onda senoidal en una bobina de rotación. La amplitud de la señal es proporcional a la intensidad del campo, siempre que sea uniforme, y al seno del ángulo entre el eje de rotación de la bobina y las líneas de campo. Este tipo de magnetómetro está obsoleto.
- Efecto Hall, son los magnetómetros más comunes. Estos sensores producen un voltaje proporcional al campo magnético aplicado y con la polaridad relativa al sentido del campo. Se utilizan en aplicaciones donde la intensidad del campo magnético es relativamente grande, tal como en los sistemas de antibloqueo de frenado en los automóviles que detectan la velocidad de rotación de la rueda a través de ranuras en los discos de las ruedas.

Los magnetómetros se caracterizan por ser muy susceptibles a errores, debido principalmente a la contaminación electromagnética [45].

6.3.4. Sensores de distancia

Sensores de distancia por infrarrojos

Estos dispositivos (ver figura (19)) emplean el método de triangulación utilizando un pequeño Sensor Detector de Posición (PSD) lineal para determinar la distancia o la presencia de los objetos dentro de su campo de visión. Básicamente su modo de funcionamiento consiste en la emisión de un pulso de luz infrarroja, que se transmite a través de su campo de visión y que se refleja contra un objeto o que, por el contrario, no lo hace.

Si no encuentra ningún obstáculo, el haz de luz no refleja, y en la lectura que se hace indica que no hay ningún obstáculo. En el caso de encontrar un obstáculo el haz de luz infrarroja se reflecta y crea un triángulo formado por el emisor, el punto de reflexión en el obstáculo y el detector, como se puede ver en la figura 20. La información de la distancia se extrae midiendo el ángulo recibido. Si el ángulo es grande, entonces el obstáculo se encuentra cerca. En el caso de que el ángulo sea pequeño, el objeto, se encontrará lejos [7].

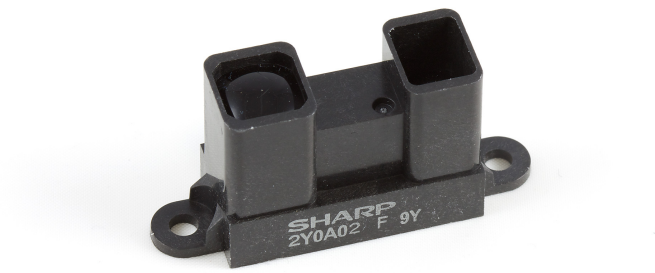

Figura 19: sensor de distancia por infrarrojos Sharp 2Y0A02

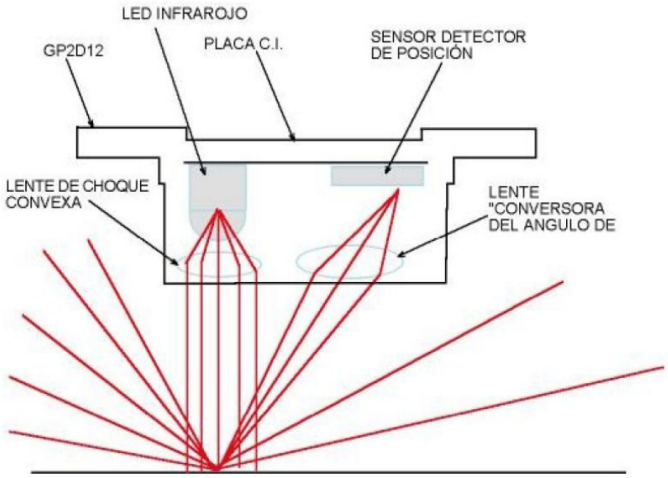

Figura 20: triangulación de un sensor de distancia por infrarrojos

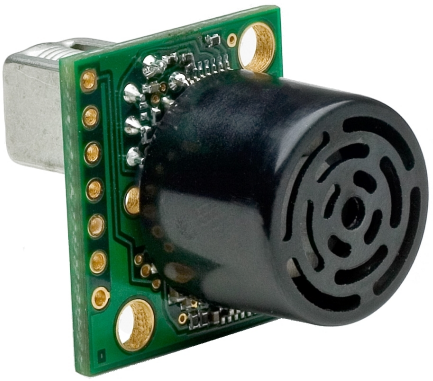

Figura 21: sensor de distancia por ultrasonidos I2C-EZ

Sensores de distancia por ultrasonidos

Estos sensores (ver figura (21)) basan su funcionamiento en el sistema que emplean los murciélagos para no chocar contra ningún objeto cuando vuelan en la oscuridad. El sensor emite pulsos de ultrasonidos. Dichos ultrasonidos rebotan contra los objetos que tienen delante y vuelven hacia el sensor. En función del tiempo que tarda el sonido ser reflejado y volver se calcula la distancia a la que se encuentra dicho objeto [7].

6.4. MATLAB

MATLAB (abreviatura de *MATrix LABoratory*, laboratorio de matrices) es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Desarrollado por MathWorks, está disponible para las plataformas Unix, Windows y Mac OS X.

MATLAB permite manipular matrices, hacer gráficas de funciones y datos, implementar algoritmos, creación de interfaces de usuario, e interfaces con programas escritos en otros lenguajes de programación entre los que se que se encuentran C, C++, Java y Fortran. Aunque MATLAB está diseñado principalmente para la computación numérica, una toolbox opcional usa el motor simbólico MuPAD, permitiendo el acceso a las capacidades de cálculo simbólico.

El paquete adicional Simulink, añade un entorno gráfico de simulación multidomino y de diseño basado en modelo de sistemas de sistemas dinámicos y sistemas empotrados [49].

6.4.1. Simulink

Simulink es un entorno para la simulación multidominio y el diseño basado en modelos para sistemas dinámicos y embebidos. Presenta un entorno gráfico interactivo y

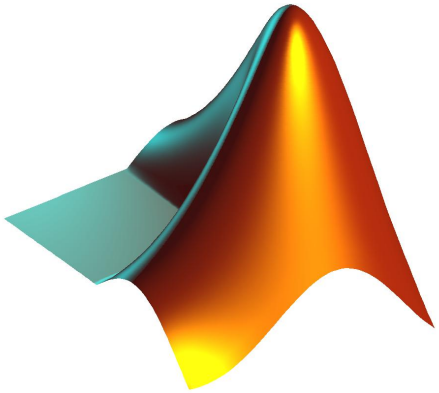

Figura 22: logotipo de MATLAB

un conjunto personalizable de bibliotecas de bloques que permiten diseñar, simular, implementar y probar una serie de sistemas variables con el tiempo, como comunicaciones, controles, procesamiento de señales, procesamiento de vídeo y procesamiento de imágenes.

Simulink está integrado con MATLAB y ofrece acceso inmediato a una amplia gama de herramientas que permiten desarrollar algoritmos, analizar y visualizar simulaciones, crear scripts de procesamiento por lotes, personalizar el entorno de modelaje y definir señales, parámetros y datos de prueba [50].

7. Métodos

En este capitulo se explica como se diseñaron las diferentes partes del sistema: modelo dinámico, unidad de medición inercial, filtro de Kalman y controlador (ver figura 23). Además se describe cómo se ha realizado la simulación mediante el software Simulink de MathWorks.

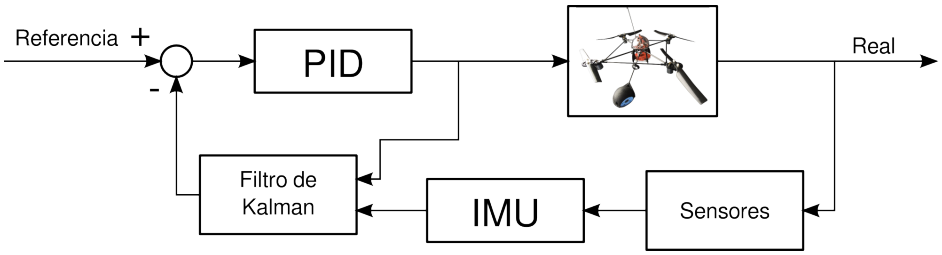

Figura 23: esquema del sistema completo

7.1. Diseño del modelo dinámico del quadrotor

Antes de comenzar con la obtención del modelo dinámico del quadrotor se realiza una introducción a los sistemas de referencia utilizados y las transformaciones para pasar de uno a otro. Después se describen las diferentes fuerza y momentos que actúan sobre el quadrotor. Esta sección concluye con la obtención del modelo dinámico completo del quadrotor.

7.1.1. Sistema de ejes

Ejes Para describir el movimiento del quadrotor respecto a un observador fijo se utiliza un marco de referencia solidario a la Tierra.

Como sistema de referencia fijo la convención NED (North, East, Down) es la más común en aplicaciones aeroespaciales. El punto de referencia o_0 se encuentra en la superficie de la Tierra y es el origen del sistema de ejes ortonormales $(o_0x_0y_0z_0)$, donde:

- $o_0x_0$apunta al norte y es tangente a los meridianos.
- $o_0y_0$apunta al este y es tangente a los paralelos.
- $o_0z_0$está en dirección al centro de la Tierra y tiene sentido hacia el centro de la Tierra como el vector gravedad.

En este trabajo se utiliza una definición más simple. El plano horizontal definido por $o_Ex_Ey_E$ será paralelo al plano formado por $o_0x_0y_0$, tangente a la superficie de la Tierra (ver figura 24). La diferencia radica en que el vector o_Ex_E apunta a una dirección

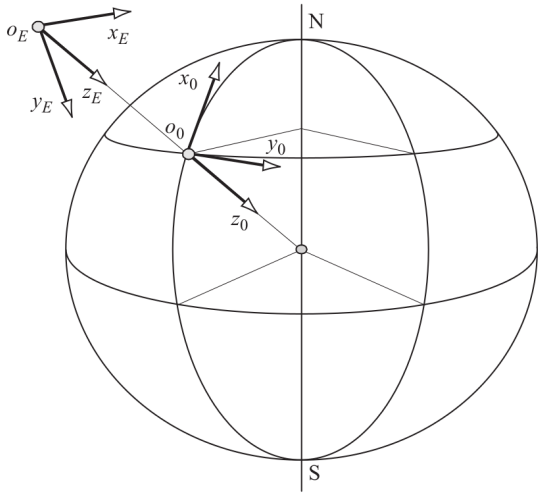

Figura 24: sistema de referencia fijo

arbitraria, no tiene por qué ser hacia el norte, pudiendo hacerlo en la dirección que más nos convenga. Por simplicidad el punto de referencia o_E coincidirá con el punto de origen del movimiento del quadrotor.

Se define también un marco de referencia móvil solidario al quadrotor. Tendrá origen en el punto o, el cual coincidirá con el centro de gravedad del quadrotor, que será el origen del sistema de ejes ortonormales $ox_by_bz_b$.

El vector ox_b está en la dirección del rotor numerado como 1, mientras que el vector oy_b apunta hacia el rotor 2 (ver figura 25). Cuando el plano formado por ox_by_b es tangente a la superficie de la Tierra la dirección de oz_b será la misma que la de la gravedad, se podría decir que ox_by_b apunta hacia el centro de la tierra.

Ángulos de Tait-Bryan Para describir la orientación del quadrotor en el espacio respecto al marco de referencia fijo se recurre a los ángulos de Tait-Bryan que son alabeo, cabeceo y guiñada, aunque son más conocidos por sus nombres en inglés (*roll*, *pitch* y *yaw* respectivamente) los cuales se utilizan de ahora en adelante.

Los tres ángulos de Tait-Bryan se definen de la siguiente manera:

- Alabeo (*roll*), es la rotación respecto al eje ox_b y está denotada por la letra griega ϕ.
- Cabeceo (*pitch*), es la rotación respecto al eje oy_b y está denotada por la letra griega θ.
- Guiñada(*yaw*), es la rotación respecto al eje oz_b y está denotada por la letra griega ψ.

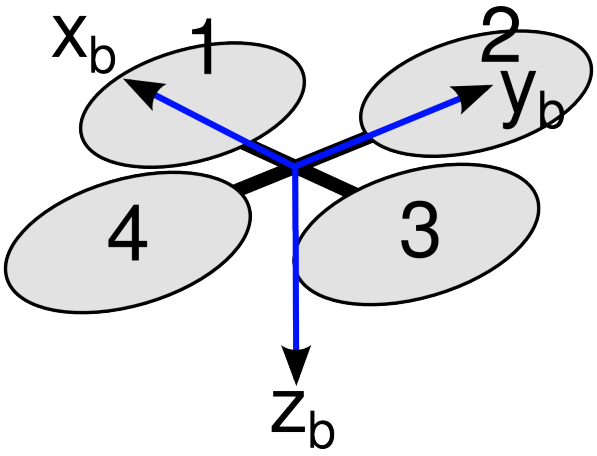

Figura 25: sistema de referencia móvil

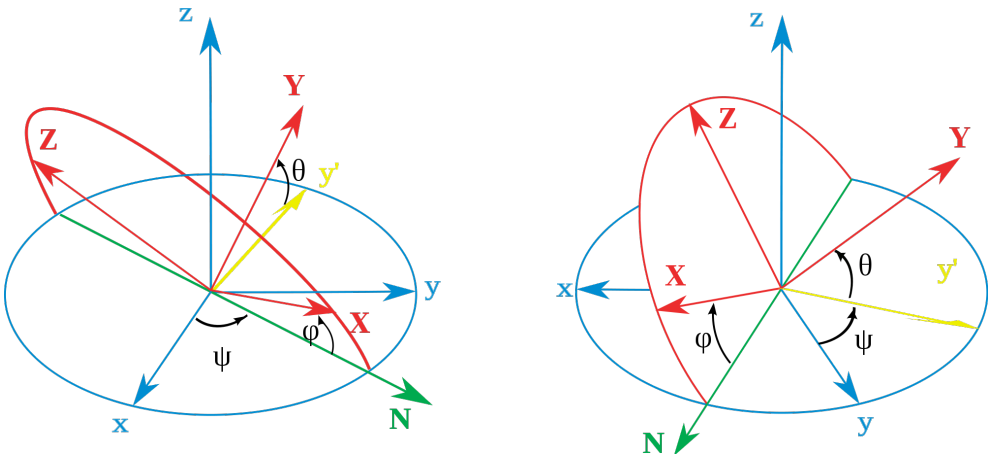

Figura 26: ángulos de Tait-Bryan

El uso de los ángulos de Tait-Bryan está muy extendido a causa de que numerosas organizaciones han publicado diversos estándares para ser seguidos debido a la especial importancia de las convenciones internacionales en vehículos aéreos. Uno de estos estándares, la norma DIN 9300 para la aeronavegación, fue adoptada por la ISO como ISO 1151-2:1985 [43].

7.1.2. Matrices de cambio de base

Transformación de cantidades lineales. Dado, por ejemplo, (ox_3, oy_3, oz_3) el cual representa componentes de una cantidad lineal en el sistema de ejes $(ox_3oy_3oz_3)$, y dado (ox_0, oy_0, oz_0), el cual representa las componentes de esa misma cantidad lineal respecto al sistema de ejes $(ox_0y_0z_0)$. Para obtener la matriz de cambio de base D se efectúa una rotación respecto a cada eje.

Se comienza girando respecto al eje ox_3 un ángulo ϕ

$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sen\phi \\ 0 & -sen\phi & cos\phi \end{bmatrix} \begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix}. \tag{7.1}$$

Se gira un ángulo θ respecto al eje oy_2

$$\begin{bmatrix} ox_2 \\ oy_2 \\ oz_2 \end{bmatrix} = \begin{bmatrix} cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} ox_1 \\ oy_1 \\ oz_1 \end{bmatrix}. \tag{7.2}$$

Por último se rota respecto al eje oz_1 un ángulo ψ

$$\begin{bmatrix} ox_1 \\ oy_1 \\ oz_1 \end{bmatrix} = \begin{bmatrix} cos\psi & sen\psi & 0 \\ -sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix}. \tag{7.3}$$

Como resultado de la sustitución recursiva en las ecuaciones (7.1), (7.2) y (7.3) se obtiene

$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix} \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix} \tag{7.4}$$

o

$$\begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} = D \begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix} \tag{7.5}$$

donde la matriz de cosenos directores D está definida por

$$D = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\phi s\theta c\psi - c\phi s\psi & s\phi s\theta s\psi + c\phi c\psi & s\phi c\theta \\ c\phi s\theta c\psi + s\phi s\psi & c\phi s\theta s\psi - s\phi c\psi & c\phi c\theta \end{bmatrix}. \tag{7.6}$$

La transformación de $(ox_3y_3z_3)$ a $(ox_0y_0z_0)$ es obtenida invirtiendo la matriz de cosenos directores D y se obtiene a través de la siguiente ecuación

$$\begin{bmatrix} ox_0 \\ oy_0 \\ oz_0 \end{bmatrix} = D^{-1} \begin{bmatrix} ox_3 \\ oy_3 \\ oz_3 \end{bmatrix} \tag{7.7}$$

donde la inversa de la matriz de cosenos directores es

$$D^{-1} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\phi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi c\phi \\ -s\theta & c\theta s\phi & c\theta s\phi \end{bmatrix} \tag{7.8}$$

Transformación de velocidades angulares. La aplicación más útil de la transformación de cantidades angulares relaciona las velocidades angulares p, q y r en los ejes del sistema de referencia móvil y las componentes de la velocidad angular, $\dot{\phi}$, $\dot{\theta}$ y $\dot{\psi}$, respecto a los ejes del sistema de referencia fijo.

Las velocidades angulares p, q y r están expresadas en los ejes del sistema de referencia móvil solidario al quadrotor $(o_Ey_Ez_E)$, por tanto, considerando cada rotación necesaria para hacer que los ejes de este marco de referencia coincidan con los ejes del sistema de referencia fijo $(ox_Ey_Ez_E)$. Primero se rota en torno a ox_E un ángulo ϕ con velocidad angular $\dot{\phi}$. Después se gira en torno a oy_E un ángulo θ con velocidad angular $\dot{\theta}$. Por último se rota en torno oz_E un ángulo ψ con velocidad angular $\dot{\psi}$. La relación entre las velocidades angulares respecto al sistema de referencia móvil y respecto al sistema de referencia fijo, se establecen con facilidad de la siguiente manera:

p es igual a la suma de las componentes de $\dot{\phi}$, $\dot{\theta}$ y $\dot{\psi}$ proyectadas sobre ox_E

$$p = \dot{\phi} - \dot{\psi} sen\theta \tag{7.9}$$

q es igual a la suma de las componentes de $\dot{\phi}$, $\dot{\theta}$ y $\dot{\psi}$ proyectadas sobre oy_E

$$q = \dot{\theta} cos\phi - \dot{\psi} sen\phi cos\theta \tag{7.10}$$

r es igual a la suma de las componentes de $\dot{\phi}$, $\dot{\theta}$ y $\dot{\psi}$ proyectadas sobre oz_E

$$r = \dot{\psi} cos\phi cos\theta - \dot{\theta} sen\phi \tag{7.11}$$

Las ecuaciones (7.9), (7.10) y (7.11) pueden combinarse en notación matricial como

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & cos\phi & sen\phi cos\theta \\ 0 & -sen\phi & cos\phi cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \tag{7.12}$$

siendo la inversa de la ecuación (7.12)

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sen\phi tg\theta & cos\phi tg\theta \\ 0 & cos\phi & -sen\phi \\ 0 & sen\phi cos\theta & cos\phi sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{7.13}$$

Cuando los ángulos ϕ, θ y ψ son pequeños las ecuaciones (7.12) y (7.13) pueden aproximarse como

$$\begin{array}{l} p = \dot{\phi} \\ q = \dot{\theta} \\ r = \dot{\psi} \end{array}. \tag{7.14}$$

7.1.3. Fuerzas

Peso. El campo gravitatorio terrestre es el causante de la fuerza peso al provocar una aceleración g de valor $9,81 \mathrm{m/s^2}$ sobre el quadrotor.

La fuerza peso puede ser modelada fácilmente respecto al marco de referencia fijo mediante la segunda ley de la dinámica de Newton

$${}^{E}\vec{F}_g = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}. \tag{7.15}$$

Se utiliza la matriz de rotación D para obtener la fuerza peso respecto al marco de referencia móvil

$${}^{b}\vec{F}_g = D\,{}^{E}\vec{F}_g = mg \begin{bmatrix} -sen\theta \\ sen\theta cos\phi \\ cos\theta cos\phi \end{bmatrix}. \tag{7.16}$$

Empuje aerodinámico. El empuje aerodinámico se genera debido a la rotación de las hélices en un fluido viscoso, aire en este caso, y es utilizado para mantener el quadrotor en el aire. El empuje se relaciona con la velocidad de rotación de los rotores mediante la ecuación

$$T = K_{empuje}\omega^2. \tag{7.17}$$

Si se considera que los rotores están perfectamente alineados con respecto al sistema de referencia móvil la dirección de la fuerza de empuje tendrá como dirección el eje z_b, siendo la componente en el resto de ejes nula.

Numerando cada rotor como $j = 1..,4$, el empuje total puede ser descrito como

$$T_z = \sum_{j=1}^{4} T_j. \tag{7.18}$$

La fuerza de empuje generada por los rotores que se mueven solidarios al marco de referencia móvil será

$${}^{b}\vec{F_T} = \begin{bmatrix} 0 \\ 0 \\ -T_z \end{bmatrix}. \tag{7.19}$$

Arrastre aerodinámico. Cuando un cuerpo sólido se mueve en un fluido, en este caso el quadrotor en el aire, se originan fuerzas causadas por la viscosidad del fluido que se oponen al este movimiento.

La fórmula general de resistencia puede deducirse mediante el método de análisis dimensional.

$${}^{E}\vec{F_d} = -\frac{1}{2}\rho A_c \begin{bmatrix} C_x & 0 & 0 \\ 0 & C_y & 0 \\ 0 & 0 & C_z \end{bmatrix} \begin{bmatrix} \dot{x}\,|\dot{x}| \\ \dot{y}\,|\dot{y}| \\ \dot{z}\,|\dot{z}| \end{bmatrix} \tag{7.20}$$

Como se puede observar observar, la fuerza de arrastre, será proporcional al cuadrado de la velocidad y tendrá su misma dirección pero sentido opuesto. C_x, C_y y C_z son coeficientes experimentales o teóricos que dependen de la forma del cuerpo, A_c, un área característica del cuerpo, y ρ, la densidad del aire.

7.1.4. Momentos

Momento debido al empuje aerodinámico. Los rotores, numerados de 1 a 4, se encuentran situados en los ejes x_b, y_b, $-x_b$ y $-y_b$ respectivamente a una distancia l del centro de gravedad del quadrotor.

La diferencia de empuje entre los rotores provocará momentos respecto a los ejes x_b e y_b del marco de referencia móvil.

$${}^{b}\vec{M_T} = \begin{bmatrix} l & 0 & 0 \\ 0 & l & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} T_4 - T_2 \\ T_3 - T_1 \\ 0 \end{bmatrix} = \begin{bmatrix} l{\cdot}(T_4 - T_2) \\ l{\cdot}(T_3 - T_1) \\ 0 \end{bmatrix} \tag{7.21}$$

Momento debido al arrastre aerodinámico. La diferencia en la velocidad de giro de los rotores generará en el eje z_b un momento de arrastre aerodinámico debido a la fricción entre el aire y las aspas de los rotores. Al girar las aspas de los rotores están sometidas a un esfuerzo causado por el rozamiento que se produce entre las aspas en movimiento y el aire lo que genera un momento de sentido contrario a la dirección de giro de los rotores.

Este momento es proporcional al empuje que genera cada rotor, siendo la constante de proporcionalidad $K_{arrastre}$, y de signo positivo si el rotor gira en sentido horario o de signo negativo si lo hace en sentido antihorario.

Los rotores pares (2 y 4) giran en el sentido de la agujas del reloj mientras que los impares (1 y 3) lo hacen en sentido opuesto, por tanto, el momento total debido al arrastre aerodinámico será

$$ {}^{b}\vec{M}_d = \begin{bmatrix} 0 \\ 0 \\ K_{arrastre}\left(T_2 + T_4 - T_1 - T_3\right) \end{bmatrix}. \tag{7.22} $$

Efecto giroscópico de los rotores. Los cuatro rotores inducen momentos debidos a efectos giroscópicos por su velocidad angular, ω_j, siendo un mecanismo adicional que hacen al quadrotor rotar respecto al eje z_b, movimiento de guiñada. El efecto giroscópico también afecta a los otros dos ejes aunque de forma menos intensa.

Si se considera un único rotor j, el momento resultante de la interacción entre el rotor y la rotación del quadrotor está dado por la ecuación

$$ {}^{b}\vec{M}_g^j = \left.\frac{dL_o}{dt}\right|_S + \Omega \times L = I\cdot\dot{\omega} + \vec{\Omega} \times (I\cdot\vec{\omega}) \tag{7.23} $$

Sustituyendo, se obtiene

$$ {}^{b}\vec{M}_g^j = \begin{bmatrix} I_{xx}\dot{\omega}_x^j \\ I_{yy}\dot{\omega}_y^j \\ I_{zz}\dot{\omega}_z^j \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} I_x\omega_x^j \\ I_y\omega_y^j \\ I_z\omega_z^j \end{bmatrix}. \tag{7.24} $$

El giro de los rotores ω_j coincide con el eje z_b por lo que las componentes ω_x^j y ω_y^j son cero. Resolviendo el producto vectorial

$$ {}^{b}\vec{M}_g^j = \begin{bmatrix} 0 \\ 0 \\ I_{zz}\dot{\omega}_j \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ I_{zz}\omega_j \end{bmatrix} = \begin{bmatrix} I_z\omega_j\dot{\theta} \\ -I_z\omega_j\dot{\phi} \\ I_z\dot{\omega}_j \end{bmatrix} \tag{7.25} $$

Si se tienen en cuenta todos los rotores, las velocidades angulares deben ser sumadas teniendo en cuenta su respectivo signo. Si además se asume que todos los rotores son

iguales y por tanto el momento de inercia de estos I_{zz}, al que se denotará a partir de ahora como J_r, es el mismo para todos definimos L_r como

$$L_r = J_r \sum_{j=1}^{4} \omega_j = J_r \left(\omega_1 - \omega_2 + \omega_3 - \omega_4\right). \tag{7.26}$$

El momento generado por efectos giroscópicos respecto al marco de referencia móvil será

$${}^{b}\vec{M}_g = \begin{bmatrix} L_r\dot{\theta} \\ -L_r\dot{\phi} \\ \dot{L}_r \end{bmatrix}. \tag{7.27}$$

7.1.5. Dinámica

Aceleraciones lineales. La segunda ley de la dinámica de Newton establece que la derivada de la cantidad de una partícula de masa constante es igual a la suma de todas las fuerzas externas que actúan sobre la partícula

$$\sum \vec{F} = \frac{d(mv)}{dt}, \tag{7.28}$$

donde m es la masa de la partícula, v es la velocidad de la partícula y su producto mv es la cantidad de movimiento.

Al ser la masa constante es equivalente a

$$\sum \vec{F} = m\frac{dv}{dt}. \tag{7.29}$$

Generalizando para un sólido rígido compuesto de tales partículas, cada una con una masa infinitesimal dm y con un vector de posición $\vec{r}$

$$\sum \vec{F} = \frac{d^2}{dt^2} \int \vec{r} dm \tag{7.30}$$

Sea M la masa total del sólido rígido, la cual es constante, se puede multiplicar y dividir por M el lado derecho

$$\sum \vec{F} = M\frac{d^2}{dt^2} \left(\frac{\int \vec{r} dm}{M} \right) \tag{7.31}$$

La expresión $\frac{\int \vec{r} dm}{M}$ es la fórmula del la posición del centro de gravedad del sólido rígido. Denotándolo como $\vec{r_{cdg}}$ la segunda ley de la dinámica de Newton quedaría como

$$\sum \vec{F} = M\vec{\ddot{r}}_{cdg} \tag{7.32}$$

La posición del quadrotor la determinaremos respecto al marco de referencia fijo, por tanto, se utiliza el sumatorio de fuerzas expresadas respecto a este marco de referencia

$$\sum {}^{E}\vec{F} = M\vec{\ddot{r}}_{cdg} \tag{7.33}$$

Aceleraciones angulares. La suma de momentos sobre el centro de gravedad de un sólido rígido debidos a fuerzas externas es igual a la derivada del momento angular en el centro de gravedad.

$$\sum \vec{M}_O = \frac{d\vec{H}_O}{dt} \tag{7.34}$$

Expresado en términos de las componentes del vector de momentos angulares, aplicando el teorema del momento cinemático, se puede escribir la ecuación como

$$\begin{gathered}\sum \vec{M}_O = \frac{dH_x}{dt}\hat{i} + \frac{dH_y}{dt}\hat{j} + \frac{dH_z}{dt}\hat{k} + \vec{\Omega} \times \vec{H} \\ \sum \vec{M}_O = [I]\,\vec{\dot{\omega}} + \vec{\Omega} \times [I]\,\vec{\omega},\end{gathered} \tag{7.35}$$

donde $\vec{\Omega}$ es la velocidad angular del sistema de coordenadas.

Si se deja el sistema de coordenadas solidario al sólido, de manera que la velocidad angular del sólido $\vec{\omega}$ sea igual que la velocidad angular Ω entonces

$$\Omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{7.36}$$

Interesa conocer la aceleración angular respecto al sistema de referencia móvil solidario al quadrotor. Para obtenerla se utiliza el sumatorio de momentos angulares respecto al sistema de referencia móvil y se despeja la aceleración angular

$$\vec{\dot{\omega}} = [I]^{-1} \sum {}^{b}\vec{M}_O - [I]^{-1}\,[\Omega] \times [I]\,\vec{\omega} \tag{7.37}$$

Sustituyendo se llega al modelo dinámico que define la actitud de quadrotor y se utilizará para el diseño del estimador

$$\begin{aligned}\ddot{\phi} &= \frac{l\cdot(T_4 - T_2)}{I_{xx}} + L_r\dot{\theta} + \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} \\ \ddot{\theta} &= \frac{l\cdot(T_3 - T_1)}{I_{yy}} + L_r\dot{\phi} + \frac{I_{zz} - I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi}\end{aligned} \tag{7.38}$$

$$\ddot{\psi} = \frac{K_{arrastre}\left(T_2 + T_4 - T_1 - T_3\right)}{I_{zz}} + \frac{I_{xx} - I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi}$$

7.2. Unidad de Medición Inercial (IMU)

La actitud del quadrotor se calculará a través de las mediciones, donde estas serán obtenidas a partir de los sensores con los que está equipado el quadrotor. El giroscopio no dará el ritmo de cambio de los diferentes ángulos de Tait-Bryan, mediante los acelerómetros se corregirá el ángulo de alabeo (ϕ) y de cabeceo (θ), y el magnetómetro se utilizara corregir la guiñada (ψ). Para realizar la fusión de los datos de los diferentes sensores se utilizará el algoritmo propuesto por en el artículo de Euston *et al.* [14].

Como se analizó en la sección dedicada a las matrices de cambio de base, en la página 52, una matriz de rotación de rotación describe la orientación de un marco de referencia respecto a otro. Las columnas de la matriz son los vectores directores unitarios de un marco respecto del otro.

$$R = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}$$

Una de las propiedades clave de las matrices de rotación es su ortogonalidad, lo que significa que si dos vectores son perpendiculares en un marco de referencia lo son en cualquier sistema de referencia. La longitud de los vectores directores también es la misma en cualquier sistema de referencia. Los errores numéricos pueden hacer que esta propiedad deje de cumplirse.

Además, la matriz de rotación, es una matriz antisimétrica lo que significa que es igual a la opuesta de su traspuesta.

El objetivo es calcular la matriz de cosenos directores sin realizar ninguna aproximación que viole la no linealidad de las ecuaciones.

Las rotaciones respecto a los diferentes ejes no son conmutativas, no se puede integrar directamente la señal de los giroscopios para obtener los ángulos. La velocidad angular de un vector debido a su rotación está dada por

$$\frac{d\vec{r}\,(t)}{dt} = \vec{\omega}\,(t) \times \vec{r}\,(t)\,. \tag{7.39}$$

La ecuación diferencial no es lineal. El vector de rotación, $\vec{\omega}\,(t)$, es multiplicado vectorialmente por la variable que intentamos integrar.

Si conocemos las condiciones iniciales y la evolución a lo largo del tiempo del vector de rotación se puede integrar numéricamente la ecuación (7.39) para obtener el vector rotado

$$\vec{r}\,(t) = \vec{r}\,(0) + \int_0^t d\vec{\theta}\,(\tau) \times \vec{r}\,(\tau) \tag{7.40}$$

$$d\vec{\theta}\,(\tau) = \vec{\omega}\,(\tau)\,d\tau$$

La estrategia será aplicar la ecuación (7.40) a la filas o columnas de la matriz de rotación R, tratándolos como vectores que giran.

El primer problema que aparece es que los vectores que se dedea seguir, y el vector de rotación, no se miden en el mismo marco de referencia. Idealmente, se desearía seguir los ejes del quadrotor en el marco de referencia fijo. Hay una solución fácil mediante el reconocimiento de la simetría en la rotación. En el sistema de referencia móvil, la rotación del marco de referencia fijo es igual y de sentido opuesto a la rotación del sistema de referencia móvil respecto al fijo. Se pueden seguir los ejes de la tierra vistos desde el sistema móvil cambiando el signo de la señal de los giroscopios. Por comodidad se pueden dejar los signos iguales e intercambiar los factores del producto vectorial.

$$\vec{r}_e\,(t) = \vec{r}_e\,(0) + \int_0^t \vec{r}_e\,(\tau) \times d\vec{\theta}\,(\tau) \tag{7.41}$$

$$d\vec{\theta}\,(\tau) = \vec{\omega}\,(\tau)\,d\tau$$

Los vectores de la ecuación (7.41) son las filas de la matriz de rotación R. A la hora de implementar la ecuación (7.41) se utiliza la misma aproximación que realizó Euston en sus artículo [14]. Se comienza volviendo a la ecuación (7.41)

$$\vec{r}_e\,(t + dt) = \vec{r}_e\,(t) + \vec{r}_e\,(t) \times d\vec{\theta}\,(t) \tag{7.42}$$

$$d\vec{\theta}\,(t) = \vec{\omega}\,(t)\,dt$$

Cuando se repite la ecuación (7.42) en cada uno de los ejes del marco de referencia fijo, se puede expresarlo matricialmente como

$$R\,(t + dt) = R\,(t) \begin{bmatrix} 1 & -\omega_z dt & \omega_y dt \\ \omega_z dt & 1 & -\omega_x dt \\ -\omega_y dt & \omega_x dt & 1 \end{bmatrix}. \tag{7.43}$$

Mientras la diagonal de la matriz de la ecuación (7.43) representa los términos de primer orden de la ecuación (7.42), el resto de elementos representan los términos de segundo orden. La ecuación (7.42) acumulará errores de redondeo numérico, deriva de los giroscopios, etc.

Los errores numéricos harán que la matriz R deje de ser ortogonal. Al proceso de mantener las condiciones de ortogonalidad renormalización. Se comienza calculando el producto escalar de las filas X e Y de la matriz, el cual debería ser cero. El resultado indicará cuánto están rotando las filas X e Y entre ellas

$$X = \begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix} \qquad Y = \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}$$

$$error = X \cdot Y = X^T Y = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \end{bmatrix} \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix} \tag{7.44}$$

Se aporta la mitad del error a cada una de las filas X e Y, y se rotan aproximadamente las filas X e Y en direcciones opuestas

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix}_{ort} = X_{ort} = X - \frac{error}{2} Y \tag{7.45}$$

$$\begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}_{ort} = Y_{ort} = Y - \frac{error}{2} X \tag{7.46}$$

El vector de la fila Z debe ser ortogonal respecto a los vectores X e Y. La manera más simple de lograr esto es mediante el producto vectorial de las filas X e Y

$$\begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{bmatrix}_{ort} = Z_{ort} = X_{ort} \times Y_{ort} \tag{7.47}$$

El último paso de la renormalización es escalar las filas de la matriz R para que sean vectores unitarios. Se divide cada elemento de la fila entre la raíz cuadrada de la suma de los cuadrados de los elementos de esa fila

$$X_{norm} = \sqrt{X_{ort} \cdot X_{ort}}$$

$$Y_{norm} = \sqrt{Y_{ort} \cdot Y_{ort}} \tag{7.48}$$

$$Z_{norm} = \sqrt{Z_{ort} \cdot Z_{ort}}$$

7.2.1. Corrección de la deriva

Aunque los giroscopios tienen un buen rendimiento, con una desviación del orden de unos pocos grados por segundo, se obtendría lo que se conoce como deriva. Se utilizan vectores de referencia para detectar la desviación de los giroscopios y proporcionar una retroalimentación negativa que corrija esta desviación. El principal requisito de los vectores de referencia de orientación es que no tengan deriva.

Los dos vectores de referencia que se utilizan los proporcionarán el acelerómetro y el magnetómetro.

El error de orientación se calcula tomando el producto vectorial del vector medido con el vector estimado por la matriz de cosenos directores. La magnitud del resultado será proporcional al seno del ángulo formado por los dos vectores y perpendicular a ambos. Representa un eje de rotación y la cantidad de rotación necesaria para que el vector medido sea paralelo al estimado. Retroalimentaremos la señal de los giroscopios a través de un controlador PI de manera que la orientación estimada siga gradualmente a los vectores de referencia y se cancele la deriva de los giroscopios.

Los acelerómetros se utilizan para la corrección del alabeo y el cabeceo y estos nos proporcionan un vector de referencia para el eje z_b.

Los acelerómetros miden además del vector gravedad las aceleraciones en el marco de referencia móvil. Por tanto, serán útiles para estimar el alabeo y el cabeceo cuando no existan aceleraciones en el marco de referencia móvil.

El vector de corrección del alabeo y cabeceo en el sistema de referencia móvil se calcula tomando el producto escalar del la fila Z de la matriz de cosenos directores con el vector unitario gravedad de referencia

$$errorAlabeoCabeceo = \begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{bmatrix} \times \begin{bmatrix} Acel_x \\ Acel_y \\ Acel_z \end{bmatrix} \tag{7.49}$$

La guiñada no puede estimarse a partir del acelerómetro, se utiliza un magnetómetro para realizar el cálculo. Para que la estimación sea correcta el eje z_b del magnetómetro debe estar alineado con el eje z_E del sistema de referencia fijo. Es necesario utilizar una matriz de rotación para lograr que los planos $x_b y_b$ y $x_e y_b$ sean paralelos, una vez alineados se puede calcular el ángulo de guiñada. La matriz de rotación será la siguiente

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sen\phi \\ 0 & -sen\phi & cos\phi \end{bmatrix} \begin{bmatrix} cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix}, \tag{7.50}$$

donde obsérvese que, a diferencia de la matriz de rotación utilizada hasta el momento, no se realiza rotación sobre el eje z.

Una vez realizada la rotación se obtiene la guiñada como

$$\psi_{mag} = arctan\left(\frac{mag_y}{mag_x}\right). \tag{7.51}$$

Se calcula la corrección de guiñada a través de la ecuación

$$correcciónGuiñadaFijo = R_{xx} sen\left(\psi_{mag}\right) - R_{yx} cos\left(\psi_{mag}\right). \tag{7.52}$$

La ecuación (7.52) nos da la corrección de guiñada en el marco de referencia fijo. Para poder corregir la deriva de los giroscopios es necesario conocer el vector en el marco de referencia móvil. Para calcularlo multiplicamos la corrección de la guiñada en el marco de referencia fijo por la fila Z de la matriz R

$$errorGuiñada = correcciónGuiñadaFijo \begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{bmatrix} \tag{7.53}$$

7.2.2. Controlador de retroalimentación PI

Cada uno de los vectores de corrección de deriva, guiñada y alabeo-cabeceo, son ponderados y usados para alimentar a un controlador PI, la salida es sumada al vector de los giroscopios y produce el vector de giroscopios corregido. Primero se calcula un promedio ponderado de la corrección total

$$correiónTotal = W_{\phi\theta}\, errorAlabeoCabeceo + W_{\psi}\, errorGuiñada \tag{7.54}$$

Se pasa la corrección por el controlador PI

$$\omega_P = K_P\, correiónTotal$$

$$\omega_I = \omega_{I\,k-1} + K_I\, dt\, correcciónTotal \tag{7.55}$$

$$\omega_{correión} = \omega_P + \omega_I$$

En este punto, el vector de corrección, se añade al vector de señal de los giroscopios y se inicia el proceso de nuevo.

Los ángulos de Tait-Bryan se obtinen a partir de la matriz de cosenos directores de la siguiente forma

$$\begin{aligned} \phi &= arctan\left(\tfrac{r_{zy}}{r_{zz}}\right) \\ \theta &= arcsen\left(r_{zx}\right) \\ \psi &= arctan\left(\tfrac{r_{yx}}{r_{xx}}\right) \end{aligned} \tag{7.56}$$

7.3. Filtro de Kalman

El control del quadrotor se realizará en actitud. El vector de estados x a utilizar contendrá los tres ángulos de Tait-Bryan así como sus derivadas.

$$x = \begin{bmatrix} \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{7.57}$$

En la ecuación (7.38) se tenía el modelo dinámico, donde ahora se escriben las ecuaciones en función de las variables de entrada τ_ϕ, τ_θ, τ_ψ y u_g. Además, se implifica la notación de los momentos de inercia siendo $I_{xyz} = \frac{I_{xx}-I_{yy}}{I_{zz}}$, $I_{yzx} = \frac{I_{yy}-I_{zz}}{I_{xx}}$ e $I_{zxy} = \frac{I_{zz}-I_{xx}}{I_{yy}}$,

$$\begin{array}{l} \ddot{\phi} = \frac{\tau_\phi}{I_{xx}} + \frac{J_r\dot{\theta}}{I_{xx}}u_g + I_{yzx}\dot{\theta}\dot{\psi} \\ \ddot{\theta} = \frac{\tau_\theta}{I_{yy}} - \frac{J_r\dot{\phi}}{I_{yy}}u_g + I_{zxy}\dot{\phi}\dot{\psi} \\ \ddot{\psi} = \frac{\tau_\psi}{I_{zz}} + I_{xyz}\dot{\phi}\dot{\theta} \end{array} \tag{7.58}$$

donde τ_ϕ, τ_θ y τ_ψ representan los momentos generados por el empuje de los motores en los diferentes ejes, siendo estos $l\cdot(T_4 - T_3)$, $l\cdot(T_1 - T_2)$ y $K_{arrastre}\cdot(T_2 + T_4 - T_1 - T_3)$, respectivamente. El término u_g en los efectos giroscópicos debido al giro de los rotores se define como $\sqrt{\frac{T_2+T_4-T_1-T_3}{K_{empuje}}}$.

El modelo dinámico obtenido en la ecuación (7.38) está gobernado por ecuaciones diferenciales no lineales. Como estimador se va a utilizar un filtro de Kalman. Al tratarse de un estimador lineal, se linealiza el modelo dinámico respecto al punto de equilibrio $x = 0$. En el punto $x = 0$ el empuje generador por todos los rotores es el mismo, en consecuencia giran a la misma velocidad, por tanto u_g será cero.

Despreciando los términos de segundo orden, estos términos corresponden a los efectos giroscópicos producidos por la rotación del quadrotor. Como la velocidad angular del quadrotor será pequeña y el objetivo es la estabilización, pueden considerarse nulos sin que lleguen a afectar a la estimación.

El modelo linealizado será el siguiente

$$\begin{array}{l} \ddot{\phi} = \frac{\tau_\phi}{I_{xx}} \\ \ddot{\theta} = \frac{\tau_\theta}{I_{yy}} \\ \ddot{\psi} = \frac{\tau_\psi}{I_{zz}} \end{array} . \tag{7.59}$$

Expresado de manera matricial queda como

$$\dot{x} = Ax + Bu, \tag{7.60}$$

donde

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.61}$$

y

$$B = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix}. \tag{7.62}$$

Como las variables son medidas en tiempo discreto, se utilizará un filtro de Kalman discreto. Se asume que el tiempo de muestreo T_s es lo suficientemente pequeño como para que la discretización de Euler sea usada de manera efectiva en este modelo. El modelo lineal discreto es

$$x_{k+1} = A_d x_k + B_d u_k, \tag{7.63}$$

con

$$A_d = \begin{bmatrix} 1 & 0 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 & T_s & 0 \\ 0 & 0 & 1 & 0 & 0 & T_s \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.64}$$

y

$$B_d = \begin{bmatrix} 0 & 0 & 0 \\ \frac{T_s}{I_{xx}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{T_s}{I_{yy}} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{T_s}{I_{zz}} \end{bmatrix} \tag{7.65}$$

El vector de entrada u_k vendrá dado por los momentos provocados por la diferencia de empuje entre los rotores

$$u_k = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}. \tag{7.66}$$

El vector de mediciones z_k estará formado los ángulos de Tait-Bryan obtenidos por la unidad de medición inercial

$$z_k = \begin{bmatrix} \phi_{IMU} \\ \theta_{IMU} \\ \psi_{IMU} \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{7.67}$$

7.4. Diseño del controlador

Para controlar la actitud del quadrotor se utilizará un controlador PID independiente para cada eje: alabeo, cabeceo y guiñada. Se utilizará un controlador PID debido a la sencillez de su algoritmo, lo que nos permitirá ejecutar el bucle de control con mayor velocidad en el Arduino, y porque es posible aplicarlo de forma general a la mayoría de sistemas de control.

A la hora de realizar el ajuste existen multitud de métodos, los dos más conocidos son los métodos de ajuste de Ziegler y Nichols los cuales se explicaron durante la revisión bibliográfica en la página 24.

Para el control del quadrotor en orientación y altitud se requieren cuatro variables de control. T_ϕ, T_θ y T_ψ se utilizarán para el control de la orientación del quadrotor, generarán los momentos, y serán la salida de los tres controladores. T_{offset} se usará para controlar la altitud de manera manual. De esta manera el empuje generado por cada uno de los motores será

$$\begin{aligned} T_1 &= T_{offset} + \tfrac{T_\theta}{2} - \tfrac{T_\psi}{4} \\ T_2 &= T_{offset} - \tfrac{T_\theta}{2} - \tfrac{T_\psi}{4} \\ T_3 &= T_{offset} - \tfrac{T_\phi}{2} + \tfrac{T_\psi}{4} \\ T_4 &= T_{offset} + \tfrac{T_\phi}{2} + \tfrac{T_\psi}{4} \end{aligned} \tag{7.68}$$

El empuje generado por los motores no puede ser negativo nunca, con el objetivo de que esto no ocurra se han limitado las señales de control mediante el uso de funciones de saturación.

7.4.1. Controlador de alabeo y cabeceo

Si se aplica la segunda ley de la dinámica de Newton a los ejes x_b e y_b, y despreciando efectos giroscópico, se obtine que las ecuaciones diferenciales que describen el giro sobre estos ejes son, respectivamente,

$$\ddot{\phi} = \frac{T_\phi L}{I_{xx}} \tag{7.69}$$

y

$$\ddot{\theta} = \frac{T_\theta L}{I_{yy}}. \tag{7.70}$$

Como se expone en el artículo de Hoffman *et al.* [17], a bajas velocidades y con perturbaciones pequeñas, como por ejemplo en vuelo en interiores, la función de transferencia que describe el comportamiento del quadrotor se aproxima a un doble integrador con un retardo de primer orden debido a la dinámica de los motores.

Si se toma la transformada de Laplace de ambos miembros de las ecuaciones (7.69) y (7.70), se supone que todas las condiciones iniciales son cero, y se añade el retardo de primer orden se llega a las siguientes funciones de transferencia

$$G(s) = \frac{\Phi(s)}{T(s)} = \frac{L}{I_{xx}s^2(Ts+1)} \tag{7.71}$$

$$G(s) = \frac{\Theta(s)}{T(s)} = \frac{L}{I_{yy}s^2(Ts+1)}. \tag{7.72}$$

En este caso, al tratarse de funciones de transferencia que poseen un doble integrador, resulta apropiada la utilización de un controlador del tipo PD.

Con el controlador PD las funciones de transferencia en lazo abierto son

$$G(s) = \frac{\Phi(s)}{T(s)} = \frac{L(K_p + K_d s)}{I_{xx}s^2(Ts+1)} \tag{7.73}$$

y

$$G(s) = \frac{\Theta(s)}{T(s)} = \frac{L(K_p + K_d s)}{I_{yy}s^2(Ts+1)}. \tag{7.74}$$

Con la realimentación quedan como

$$G(s) = \frac{\Phi(s)}{T(s)} = \frac{L(K_p + K_d s)}{I_{xx}s^2(Ts+1) + L(K_p + K_d s)} \tag{7.75}$$

y

$$G(s) = \frac{\Theta(s)}{T(s)} = \frac{L(K_p + K_d s)}{I_{yy}s^2(Ts+1) + L(K_p + K_d s)}. \tag{7.76}$$

7.4.2. Controlador de guiñada

Si se aplica la segunda ley de la dinámica de Newton al eje z_b y despreciando efectos giroscópico obtenemos que la ecuación diferencial que describe el giro sobre este eje es

$$\ddot{\psi} = \frac{T_\psi K_{arrastre}}{I_{zz}}. \tag{7.77}$$

Al igual que en los controladores de alabeo y cabeceo, la función de transferencia que describe el comportamiento del quadrotor en el movimiento de guiñada se aproxima a un doble integrador con un retardo de primer orden debido a la dinámica de los motores.

Si se toma la transformada de Laplace de ambos miembros de la ecuación (7.77), se supone que todas las condiciones iniciales son cero, y se añade el retardo de primer orden se llega a la siguiente función de transferencia

$$G(s) = \frac{\Psi(s)}{T(s)} = \frac{K_{arrastre}}{I_{zz}s^2(Ts+1)}. \tag{7.78}$$

Como se puede observar, la función de transferencia (7.78), es muy similar a las de alabeo y cabeceo. Como se trata de una función de transferencia que posee un doble integrador, resulta apropiado un controlador del tipo PD.

Con el controlador PD la función de transferencia en lazo abierto es

$$G(s) = \frac{\Psi(s)}{T(s)} = \frac{K_{arrastre}(K_p + K_d s)}{I_{zz}s^2(Ts+1)}. \tag{7.79}$$

Con la realimentación quedan como

$$G(s) = \frac{\Psi(s)}{F(s)} = \frac{K_{arrastre}(K_p + K_d s)}{I_{zz}s^2(Ts+1) + K_{arrastre}(K_p + K_d s)}. \tag{7.80}$$

7.5. Modelado en Simulink

El modelador y simulación del sistema completo se realizó en Simulink. Para el modelado en Simulink se dividió el sistema completo en cinco subsistemas principales:

- Modelo dinámico del quadrotor, a partir del empuje generado por los rotores determina el comportamiento dinámico del quadrotor.
- Simulación de los sensores, la salida de este subsistema recrea la señal de los sensores con los que está equipado el quadrotor real.
- Unidad de medición inercial, a partir de la señal de los sensores obtiene una aproximación de la orientación del quadrotor.

- Filtro de Kalman, con la salida de la unidad de medición inercial y un modelo dinámico linealizado del comportamiento del quadrotor. El filtro de Kalman, estima la orientación del quadrotor.
- Controlador PD, determina la señal de control para que el quadrotor se oriente de la manera deseada.

Las variables de entrada de la simulación son los tres ángulos de Tait-Bryan y el *offset* de los motores, como se explica en el diseño del controlador en la página 66.

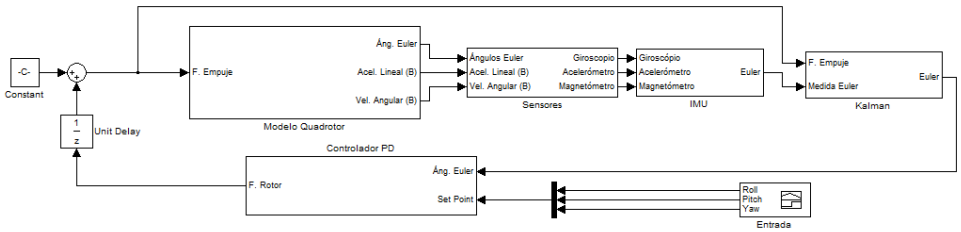

Figura 27: modelo del sistema completo en Simulink

Como se ve en la figura 27, todos los subsistemas forman un lazo cerrado. Para evitar que se produjese un bucle algebraico durante la resolución se añadió un retraso unidad a la salida del controlador tal y como se recomienda en el manual de ayuda del programa.

Las constantes en vez de estar definidas en cada uno de los bloques, se asignan desde un archivo .m (listado de programa 1), ejecutado al comienzo de la simulación, donde están definidas todas ellas. De esta manera es más sencillo modificar parámetros de la simulación.

```
g = 9.81;          % Aceleracion de la gravedad [m/s2]

masa = 2.85;       % Masa del quadrotor [kg]
L = .2;                    % Distancia entre los rotores y el cdg
    [m]
T_inercia = [.05 0 0;   % Tensor de inercia
             0 .05 0;   % del quadrotor
             0 0 .15]; % [kg/m^2]
I_rotor = 6*10^-5;       % Momento de inercia del rotor
K_empuje = 0.2*10^-3;   % Relación entre la velocidad del rotor
     y el empuje que genera [N/w^2]
K_arrastre = .05;        % Relación entre el empuje y el momento
     en un rotor

Ini_pos = [0 0 0]';      % Coordenadas iniciales del cdg del
    quadrotor
Ini_ang = [0 0 0]';      % Orientación inicial del quadrotor
```

```
14
15 Kp = 70;
16 Kd = 5;
17 Kp_yaw = 100;
18 Kd_yaw = 22;
```

Listado de programa 1: variables de la simulación

7.5.1. Modelo dinámico del quadrotor

El subsistema dedicado a determinar el comportamiento del quadrotor, figura 28, tiene como entrada la fuerza de empuje generada por cada uno de los rotores. Como salida tiene la aceleración lineal y la velocidad angular respecto al sistema de referencia móvil, así como los ángulos de Euler.

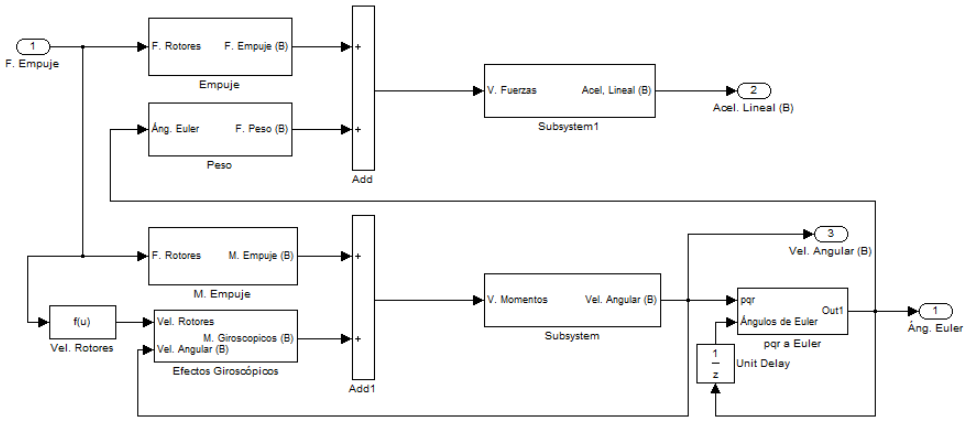

Figura 28: subsistema modelos dinámico

Está formado a su vez por siete subsistemas diferentes. El primero de ellos, figura 29, calcula el vector de la fuerza de empuje respecto al sistema de referencia móvil.

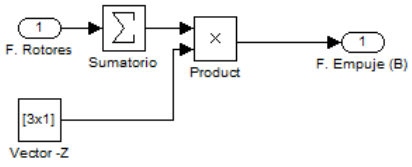

Figura 29: subsistema fuerza empuje

El vector de la fuerza peso se calcula con respecto al sistema de referencia fijo multiplicando la masa de quadrotor por la aceleración de la gravedad. Después este vector es

rotado para obtener sus componentes respecto al sistema de referencia móvil solidario al quadrotor (ver figura (30)).

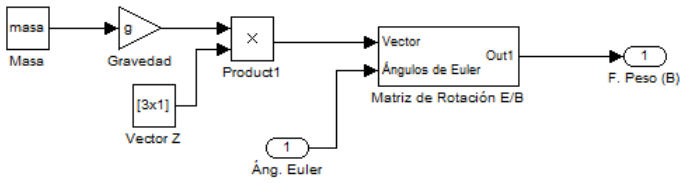

Figura 30: subsistema fuerza peso

El subsistema dedicado a calcular la aceleración lineal del centro de gravedad del quadrotor, figura 31, aplica la segunda ley de la dinámica de Newton. Se divide el vector fuerzas, obtenido al sumar la fuerza generada por el empuje de los rotores y la fuerza peso, entre la masa del quadrotor.

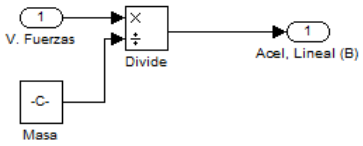

Figura 31: subsistema aceleraciones

Los momentos causados por diferencia de empuje generado por los diferentes rotores se calculan en el subsistema de la figura 32.

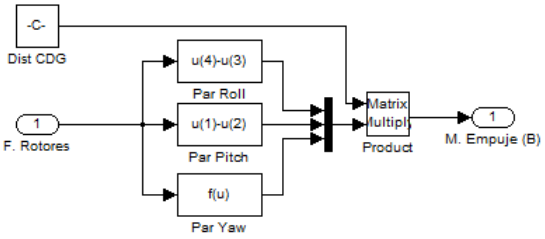

Figura 32: subsistema momentos generados por el empuje

Los efectos giroscópicos producidos por el giro de los rotores se calculan en otro subsistema, figura 33.

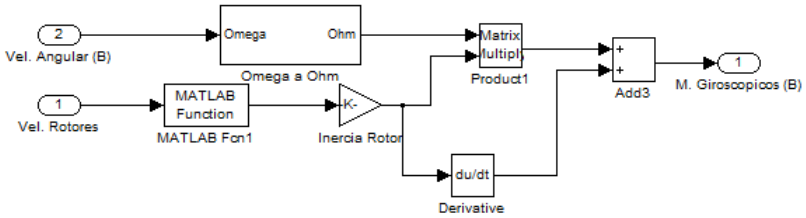

Figura 33: subsistema efectos giroscópicos

El subsistema encargado de calcular la aceleración angular, figura 34, es el más complejo de los que componen la simulación de la dinámica del quadrotor. En él se realiza el cálculo de la ecuación diferencial (7.37). La entrada del subsistema está compuesta por el vector momentos, obtenido al sumar los momentos producidos por las diferencias de empuje y los producidos por efectos giroscópicos.

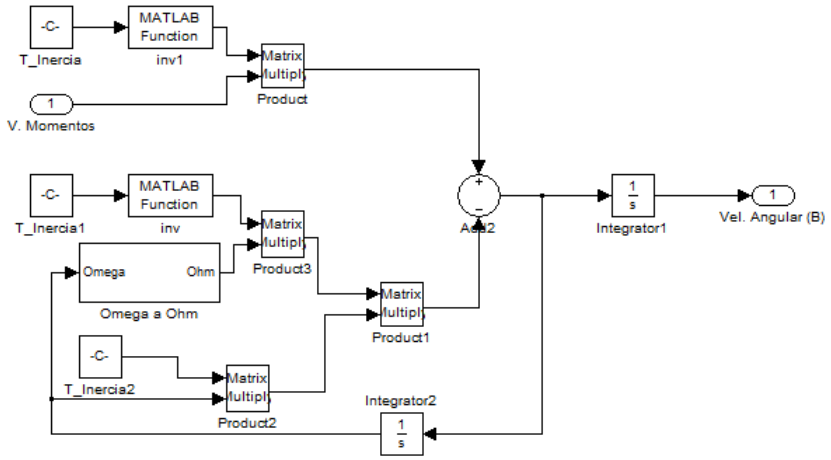

Figura 34: subsistema velocidad angular

7.5.2. Sensores

Este subsistema, a partir de la posición real del quadrotor, estima la salida de los sensores con los que esta equipado el quadrotor real: acelerómetro, giroscopio y magnetómetro (ver figura (35)). Se añade el ruido blanco adictivo y en el caso de los giroscopios una desviación. Las características de este ruido han sido obtenidas experimentalmente a partir de muestras de mediciones de los sensores.

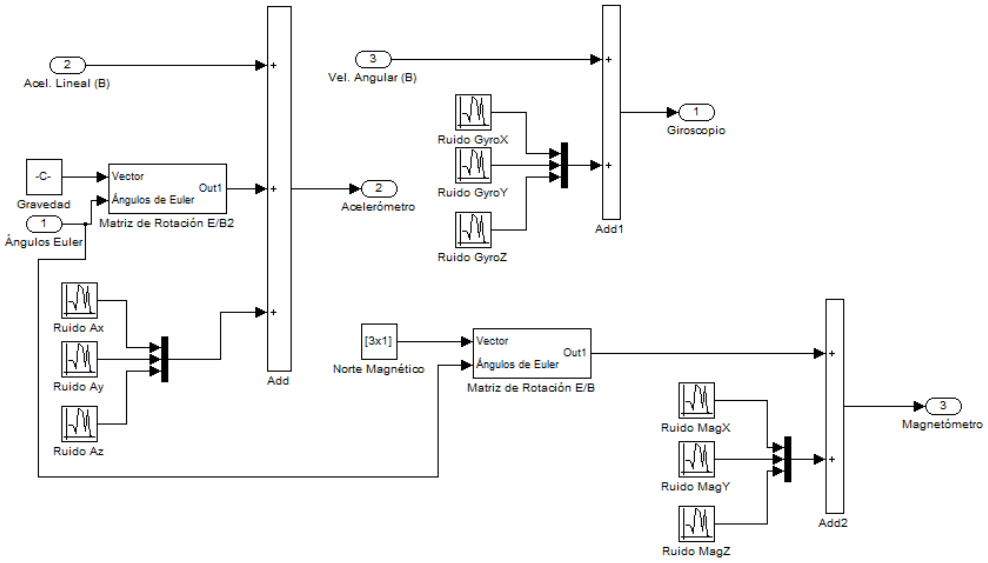

Figura 35: subsistema sensores

7.5.3. Unidad de medición inercial

La unidad de medición inercial es una implementación del algoritmo descrito en anteriormente y realizada mediante una función en lenguaje de programación M. La función es llamada con una frecuencia de 100 Hz.

En la primera ejecución se inicializan las variables que serán utilizadas en las siguientes llamadas a la función. Estas variables se almacenan como variables del tipo *persistent*. Las variables de este tipo son locales a la función que las ha declarado y su valor es almacenado entre llamadas a la función. Se diferencian de las variables globales en que son solamente conocidas por la función que las ha declarado (ver listado 2).

```
1  %
2  %       IMU BASADA EN MATRIZ DE COSENOS DIRECTORES
3  %
4  %       https://gentlenav.googlecode.com/files/DCMDraft2.pdf
5  %
6
7  function Euler = DCM(u)
8
9  giro = u(1:3);
10 acel = u(4:6);
11 mag = u(7:9);
12
13 persistent R W_I drift dt
14
15 IMU_Kp = .1;
```

```
16 IMU_Ki = .0001;
17
18 IMU_Kp_yaw = 1.2;
19
20 % Primera iteración. Inicializamos variables
21 if isempty(R)
22     R = eye(3);
23     W_I = zeros(3,1);
24     drift = [0 0 0]';
25     dt = .01;
26
27 else
28         % Actualizo R con los datos del giroscópio
29         giro = [giro(1) -giro(2) giro(3)]' + drift;
30         R_dt = [0  -giro(3)*dt giro(2)*dt; giro(3)*dt 0 -giro
               (1)*dt; -giro(2)*dt giro(1)*dt 0];
31         R = R+R*R_dt;
32
33         % Calculo el error
34         X = R(:,1);
35         Y = R(:,2);
36         error = X'*Y;
37
38         % Corrijo errores de ortogonalidad
39         X_orth = X-(error./2)*Y;
40         Y_orth = Y-(error./2)*X;
41         Z_orth = cross(X_orth,Y_orth);
42
43         % Normalizo
44         X_norm = X_orth./sqrt(sum(X_orth.^2));
45         Y_norm = Y_orth./sqrt(sum(Y_orth.^2));
46         Z_norm = Z_orth./sqrt(sum(Z_orth.^2));
47
48         % Saco X corregida
49         R = [X_norm Y_norm Z_norm];
50
51         % Error en Pitch y Roll a partir del acelerómetro
52         PitchRollError = cross(R(:,3),acel);
53
54         % Error en Yaw a partir del magnetómetro
55         Euler_prev = [atan2(R(3,2),R(3,3)) asin(R(3,1))]';
56         CMx = mag(1)*cos(Euler_prev(1)) + mag(2)*sin(
               Euler_prev(2))*sin(Euler_prev(1)) + mag(3)*cos(
               Euler_prev(2))*sin(Euler_prev(1));
57         CMy = mag(2)*cos(Euler_prev(2)) - mag(3)*sin(
               Euler_prev(2));
58
59         Mag_heading = atan(CMy/CMx);
60         errorCourse = R(1,1)*sin(Mag_heading)-R(2,1)*cos(
               Mag_heading);
61         YawError = errorCourse.*R(:,3);
62
63         % Calculamos el error total como la suma de ambos
               ponderados.
64
65         % Peso de la aceleración
66     if abs(atan2(R(3,2),R(3,3))) > .1745 ||  abs(asin(R(3,1)))
           > .1745
```

```
67          AccelWeight = 0;
68      elseif 1 - 10*abs(1 - (sqrt(sum(acel.^2))/9.81)) < 0
69                  AccelWeight = 0;
70          else
71                  AccelWeight = 1 - 10*abs(1 - (sqrt(sum(acel
                        .^2))/9.81));
72          end
73
74          TotalError = PitchRollError*AccelWeight+YawError;
75
76          % Controlador PI para corregir la desviación
77
78          W_P = IMU_Kp*PitchRollError*AccelWeight + IMU_Kp_yaw*
                    YawError;
79          W_I = W_I + IMU_Ki*dt*TotalError;
80          drift = W_P + W_I;
81
82  end
83
84  % Paso de la matriz de cosenos directores a ángulos de Euler
85  Euler = [atan2(R(3,2),R(3,3)) asin(R(3,1)) atan2(R(2,1),R(1,1)
        )]';
```

Listado de programa 2: algoritmo de la unidad de medición inercial

7.5.4. Filtro de Kalman

Como era el caso de la unidad de medición inercial, para la implementación del filtro de Kalman, se ha recurrido al uso de una función en lenguaje M. Esta función también es ejecutada con una frecuencia de 100 Hz (ver listado 3).

En la primera llamada se definen las diferentes matrices que forman parte de este algoritmo recursivo. Éstas se almacenan en variables de tipo *persistent* para que se encuentren disponibles en la siguiente llamada a la función.

```
1   %
2   %       FILTRO KALMAN
3   %
4
5   function x_est = Kalman(u)
6
7   persistent x A B H P Q R dt
8
9   QuadrotorVar;
10
11  % Almacenamos los datos correctamente
12  par_roll = (u(4)-u(3))*L;
13  par_pitch = (u(1)-u(2))*L;
14  par_yaw = (-(u(1)+u(2))+((u(3)+u(4))))*K_arrastre;
15  roll = u(5);
16  pitch = u(6);
17  yaw = u(7);
18
19  if isempty(P)
```

```
20
21      % El estado inicial es
22      x = [0 0 0 0 0 0]';
23
24      % Paso
25      dt = 0.01;
26
27      % Inicializamos la covarianza estimada
28      P = eye(6);
29
30      % A y B representan el modelo
31      A = [1  0  0 dt  0  0
32           0  1  0  0 dt  0
33           0  0  1  0  0 dt
34           0  0  0  1  0  0
35           0  0  0  0  1  0
36           0  0  0  0  0  1];
37
38       B = [0 0 0 0 0 0
39            0 0 0 0 0 0
40            0 0 0 0 0 0
41            0 0 0 dt/T_inercia(1,1) 0 0
42            0 0 0 0 dt/T_inercia(2,2) 0
43            0 0 0 0 0 dt/T_inercia(3,3)];
44
45      % H representa la matriz de medición
46      H = [1 0 0 0 0 0
47           0 1 0 0 0 0
48           0 0 1 0 0 0
49           0 0 0 0 0 0
50           0 0 0 0 0 0
51           0 0 0 0 0 0];
52
53      % Q y R son las matrices de covarianza del proceso y de
            las mediciones
54      Q = [1e-5 0 0 0 0 0
55           0 1e-5 0 0 0 0
56           0 0 1e-5 0 0 0
57           0 0 0 1e-8 0 0
58           0 0 0 0 1e-8 0
59           0 0 0 0 0 1e-8];
60
61      R = [.01  0   0  0 0 0
62            0  .01  0  0 0 0
63            0   0  .01 0 0 0
64            0   0   0  1 0 0
65            0   0   0  0 1 0
66            0   0   0  0 0 1];
67  else
68      % Filtro Kalman
69
70      % Etapa de prediccion
71      x = A*x + B*[0 0 0 par_roll par_pitch par_yaw]';
72      P  =A*P*A' + Q;
73
74      % Esto es la medida obtenida de la IMU
75      z=[roll pitch yaw 0 0 0]';
76
```

```
77      % Etapa de actualización
78      K = P*H'*pinv(H*P*H'+R);
79      x = x + K*(z-H*x);
80      P = (eye(6)-K*H)*P;
81  end
82
83  x_est = [x(1) x(2) x(3)]';
```

Listado de programa 3: algoritmo del filtro de Kalman

7.5.5. Controlador

El subsistema dedicado al controlador esta formado a su vez por tres subsistemas idénticos, cada uno con un controlador PD, y una función que calcula el empuje de cada motor en función de la salida de los controladores (ver figura 36).

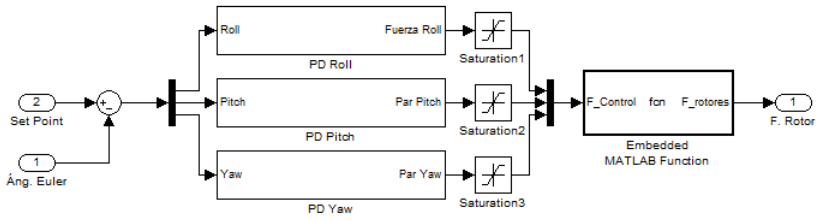

Figura 36: subsistema controlador

Los controladores PD de alabeo y cabeceo, figura 37, comparten las ganancias del controlador (K_p y K_d) mientras que las del controlador PD de guiñada son diferentes.

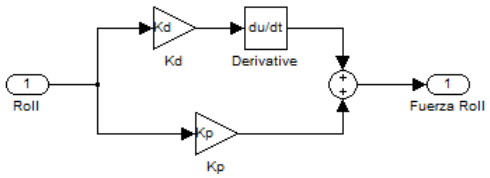

Figura 37: subsistema controlador PD alabeo

Parte IV

Resultados y discusión

8. Resultados y discusión

Para evaluar el desempeño del sistema se realizaron diferentes ensayos sobre cada uno de los componentes. Se comienza por la unidad de medición inercial, para continuar con el filtro de Kalman y el controlador PD.

En este capítulo se exponen los resultados de los ensayos realizados, así como el procedimiento usado para realizar el ajuste de los diferentes componentes del sistema.

8.1. Unidad de medición inercial

8.1.1. Corrección de deriva

En el algoritmo de la unidad de medición inercial, la corrección de la deriva en los ángulos de alabeo y cabecero se realiza a partir de la medición del vector gravedad realizada por el acelerómetro. El acelerómetro, además de medir el vector gravedad, mide las aceleraciones lineales a las que está sometido el quadrotor (ecuación (6.1)), lo que afecta a la corrección.

En la gráfica de la figura 38 se observa cómo la aceleración a la que se ve sometido el quadrotor cuando el ángulo de alabeo varía y provoca un desplazamiento, con su correspondiente aceleración, afecta a la estimación realizada por el acelerómetro.

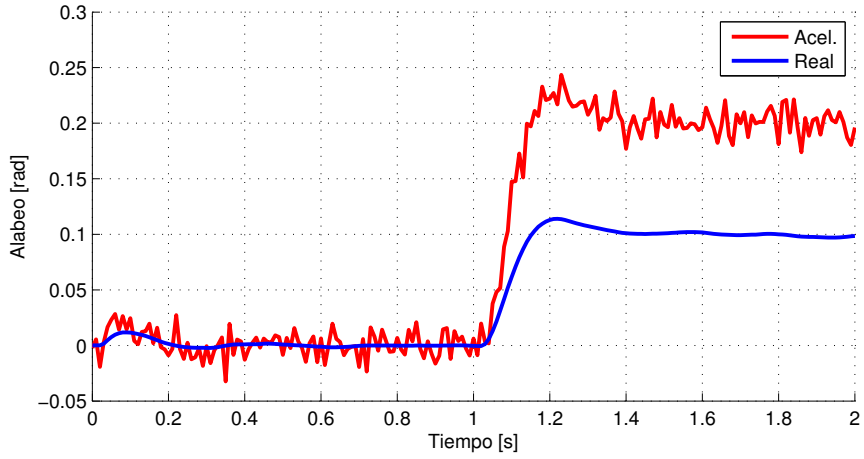

Figura 38: estimación del alabeo mediante el acelerómetro

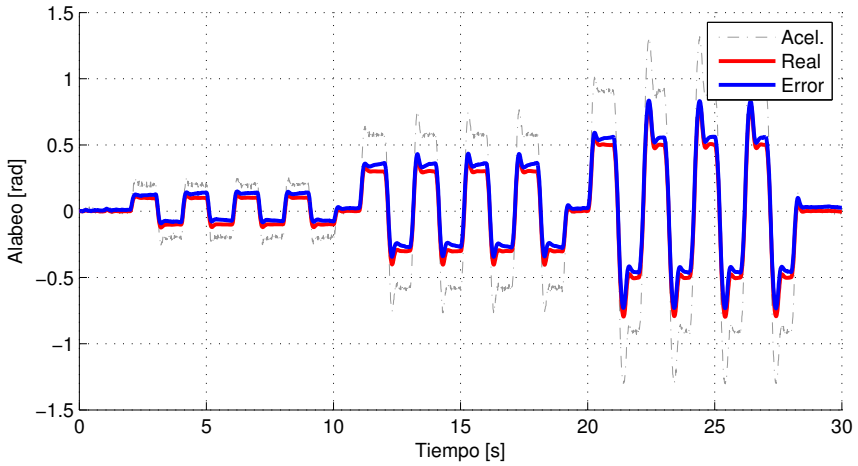

Figura 39: estimación del alabeo de la IMU con la ponderación de la corrección original

En el algoritmo original de la unidad de medición inercial, la ponderación del error en alabeo y cabeceo ($W_{\phi\theta}$), se calcula a partir del módulo de la aceleración medida por el acelerómetro. Cuando esta es mayor que 1,5 g o menor 0,5 g se deja de realizar la corrección. Es estos casos el valor de $W_{\phi\theta}$ es 0. Entre esos dos valores se calcula el peso de la corrección como

$$W_{\phi\theta} = 1 - 2 \cdot \left| \frac{\|\vec{a}\|}{9,81} \right| . \tag{8.1}$$

Aplicando este algoritmo en la unidad de medición inercial de la simulación se obtiene la gráfica de la figura 39, siendo el error de la estimación el que se puede ver en la gráfica de la figura 40. En ella se observa cómo por acción del controlador PI el valor estimado por el algoritmo se va alejando del estimado en un primer momento por los giroscopios para acercarse al valor estimado por el acelerómetro. El valor del alabeo estimado por el acelerómetro, como se ha visto anteriormente, no es correcto debido a la influencia de las aceleraciones a las que se ve sometido el quadrotor. Las características del error de la estimación en alabeo realizada mediante el algoritmo original se encuentran en la tabla 9.

El algoritmo original está diseñado para aeronaves de ala fija donde, a partir de la medición de la velocidad realizado por el sistema de posicionamiento GPS, se calcula la aceleración centrípeta a la que se ve sometida la aeronave y se resta de la medida obtenida por el acelerómetro. De este modo, quedando sólo el vector gravedad, la estimación de la orientación realizada por el acelerómetro es mejor.

Como el objetivo de este proyecto es lograr el vuelo estático y estable del quadrotor, hacemos que se deje de corregir alabeo y cabeceo cuando el quadrotor rota mas de 10°, que son los ángulos a partir de los cuales las aceleraciones debido al desplazamiento son suficientes como para generar errores en la estimación. Con este nuevo método

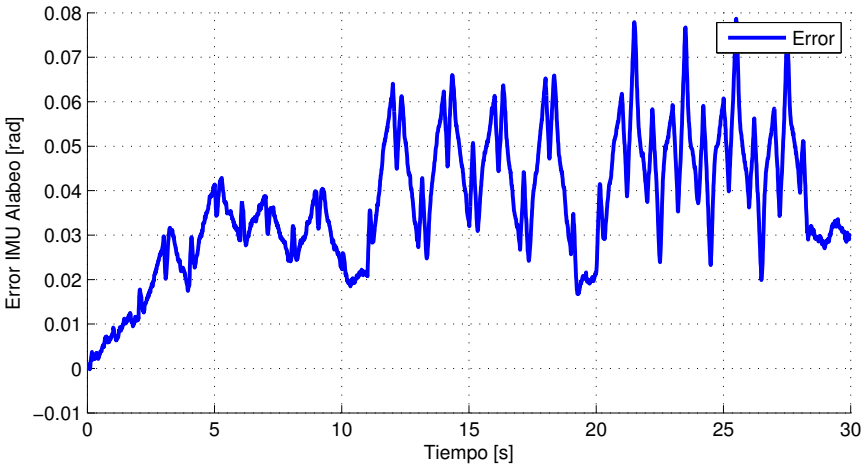

Figura 40: error en la estimación del alabeo de la IMU con la ponderación de la corrección original

Tabla 9: características de la estimación de alabeo por la IMU con la ponderación de la corrección original

Error cuadrático medio	0,0400 rad
Error medio	0,0371 rad
Error máximo	0,0787 rad

de ponderación el resultado de la unidad de medición inercial mejora, aunque deja de corregirse la deriva de los giroscopios durante el tiempo que el quadrotor se encuentra fuera de ese rango.

8.1.2. Respuesta en alabeo

Para validar la estimación del ángulo de alabeo y cabeceo realizada por la unidad de medición inercial, el sistema, fue expuesto a un tren de pulsos de amplitud creciente. Durante los 3 primeros segundos el ángulo de alabeo es 0 radianes, a partir de ese momento se introduce un tren de pulsos compuesto por 4 pulsos con un valor máximo de 0,1 radianes y un mínimo de -0,1 radianes. Tras el primer tren de pulsos, durante 1 segundo, se vuelve a la posición de equilibrio para, a continuación, repetir el tren de pulsos con valores de 0,3 y -0,3 radianes y por último con valores de 0,5 y -0,5 radianes.

En las gráficas de las figuras 41 y 42 se aprecia como la fuente del error varía en función del ángulo de alabeo a causa del algoritmo de ponderación de la corrección del alabeo y del cabeceo. Durante los primeros pulsos de 0,2 radianes de amplitud, el controlador PI provoca un error en el ángulo de alabeo al intentar seguir al ángulo estimado por el acelerómetro, el cual no es correcto. En los pulsos de 0,6 radianes y

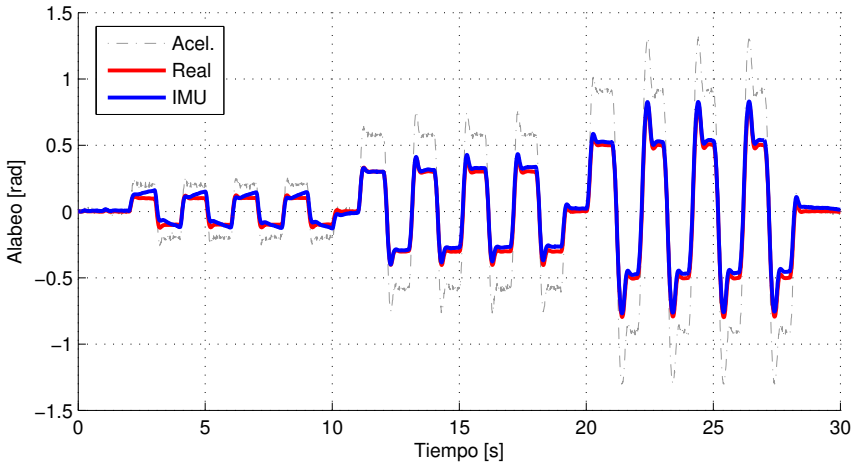

Figura 41: estimación del alabeo realizada por la IMU

Tabla 10: características de la estimación del alabeo realizada por la IMU

Error cuadrático medio	0,0281 rad
Error medio	0,0203 rad
Error máximo	0,0659 rad

1 radian de amplitud, el controlador PI deja de realizar la corrección, pasando a ser la deriva del giroscopio la principal fuente de error.

En la tabla 10 se realiza un resumen de las características de la estimación del ángulo de alabeo realizada por la unidad de medición inercial.

8.1.3. Respuesta en guiñada

La entrada a la que se sometió al sistema para determinar el rendimiento en la estimación realizada por la unidad de medición inercial de la guiñada es similar a la usada en alabeo-cabeceo, pero en este caso los pulsos reducen su número y frecuencia. El sistema comienza con 3 segundos a 0 radianes para ser sometido a un tren de impulsos de amplitud creciente de amplitud 0,2, 0,6 y 1 radianes con un periodo de 8 segundos cada pulso.

Como se puede apreciar en las gráficas de las figuras 43 y 44, el error en la estimación del ángulo de guiñada es pequeño en comparación con el que se tiene en la estimación del alabeo y el cabeceo. Además, al contrario que estos, el error no varía con la amplitud del pulso al que se ve sometido el sistema.

En la tabla 11 se realiza un resumen de las características de la estimación del ángulo de alabeo realizada por la unidad de medición inercial.

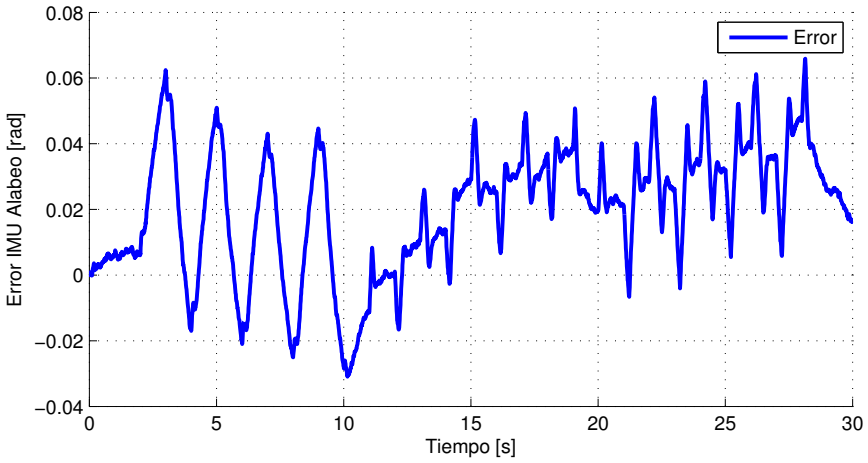

Figura 42: error en la estimación del alabeo realizada por la IMU

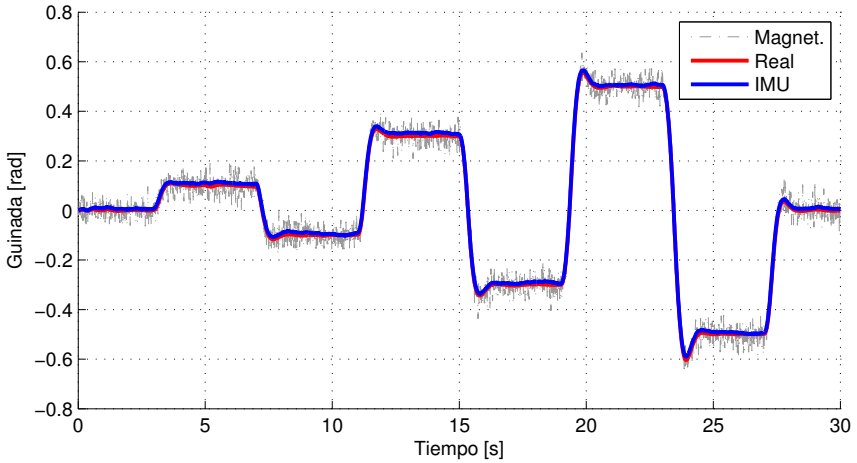

Figura 43: estimación del guiñada realizada por la IMU

Tabla 11: características de la estimación del guiñada realizada por la IMU

Error cuadrático medio	0,0157 rad
Error medio	0,0080 rad
Error máximo	0,0086 rad

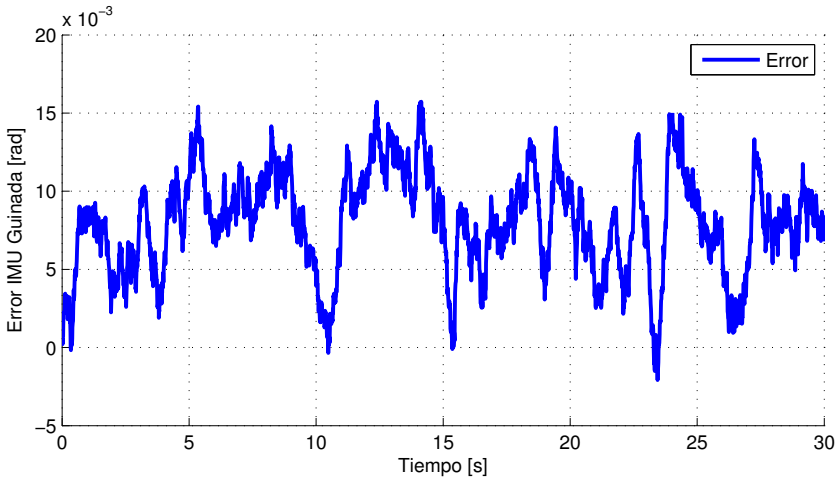

Figura 44: estimación del guiñada realizada por la IMU

8.2. Filtro de Kalman

Las pruebas utilizadas para determinar el desempeño del filtro de Kalman son las mismas que se realizaron a la unidad de medición inercial. Además, con el fin de comprobar los efectos de las no linealidades sobre el estimador lineal, se realizo otro ensayo en el que se combinaba un movimiento en alabeo con otro en guiñada.

Para que el error en la estimación de la orientación realizado por la unidad de medición no afectase a los resultados del filtro de Kalman, se usó como vector medida la orientación real del quadrotor a la que se le añadió ruido blanco aditivo de varianza 0,1.

8.2.1. Ajuste del filtro de Kalman

Los valores de la matriz de covarianza del error de la medición se determinaron a partir de la varianza de la estimación de la orientación realizada por la unidad de medición inercial respecto a la orientación real del quadrotor, obteniéndose la siguiente matriz de varianza del error de la medición R

$$R = \begin{bmatrix} 0,01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8.2}$$

La obtención de la matriz de covarianza del error del proceso es mucho más difícil de realizar. Para su determinación se recurrió a la minimización de una función de coste

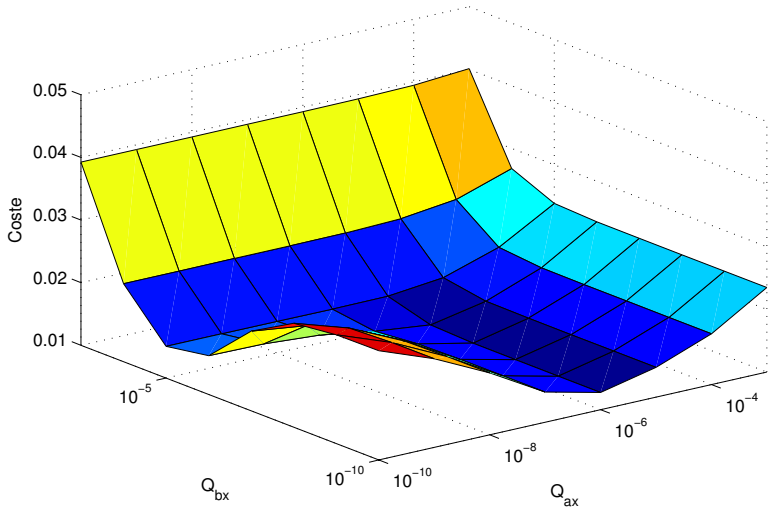

Figura 45: valor de la función coste en el eje x para diferentes valores de Q_{ax} y Q_{bx}

usando datos de una simulación previa como juego de entrenamiento. La función de coste escogida fue el error cuadrático medio, que viene dado por

$$error_{RMS} = \sqrt{\frac{\sum (x_{est\,i} - x_{real\,i})^2}{N}} \tag{8.3}$$

Los valores a determinar de la matriz de covarianza del error Q eran

$$Q = \begin{bmatrix} Q_{ax} & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_{ay} & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_{az} & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_{bx} & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_{by} & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_{bz} \end{bmatrix}. \tag{8.4}$$

En la gráfica de la figura 45 se muestra el valor de la función coste en el eje x para diferentes valores de Q_{ax} y Q_{bx}, se puede observar como el mínimo se obtiene con valores de covarianza de 10^{-7} y 10^{-6} respectivamente.

La matriz de covarianza del error del proceso resultante fue la siguiente.

$$Q = \begin{bmatrix} 10^{-7} & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^{-7} & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^{-6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{-6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^{-6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-8} \end{bmatrix} \tag{8.5}$$

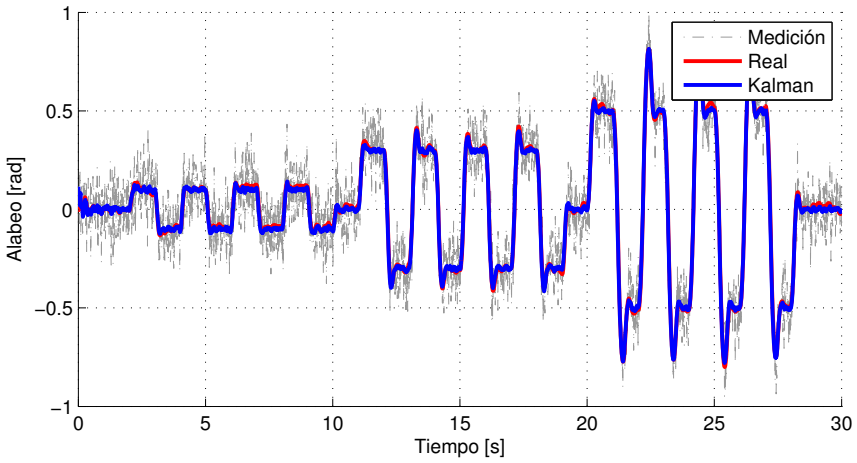

Figura 46: estimación del alabeo del filtro de Kalman

Tabla 12: características de la estimación del alabeo realizada por el filtro de Kalman

Error cuadrático medio	0,0154 rad
Error medio	-0,0046 rad
Error máximo	0,1153 rad

8.2.2. Respuesta en alabeo y cabeceo

Con el objetivo de validar el filtro de Kalman en la estimación del ángulo de alabeo y cabeceo, el sistema fue expuesto a un tren de pulsos de amplitud creciente. Durante los 3 primeros segundos el ángulo de alabeo es 0 radianes, a partir de ese momento se introduce un tren de pulsos compuesto de 4 pulsos con un valor máximo de 0,1 radianes y un mínimo de -0,1 radianes. Tras el pulso durante 1 segundo se vuelve a la posición de equilibrio para, a continuación, repetir el tren de pulsos con valores de 0,3 y -0,3 radianes y por último con valores de 0,5 y -0,5 radianes.

Como se observa en la gráfica de la figura 46, la estimación se puede considerar bastante buena. En la gráfica de la figura 47 se observa como el error alcanza su valor máximo durante la inicialización. El error durante los pulsos de amplitud 0,2 radianes varia entre los 0,02 y -0,02 radianes para ir aumentando a medida que lo hace la amplitud de los pulsos debido al aumento del peso de las no linealidades.

En la tabla 12 se realiza un resumen de las características de la estimación de ángulo de alabeo realizada por el filtro de Kalman.

8.2.3. Respuesta en guiñada

La entrada a la que se sometió al sistema para determinar el rendimiento en la estimación realizada por el filtro Kalman de la guiñada es similar a la usada en alabeo-

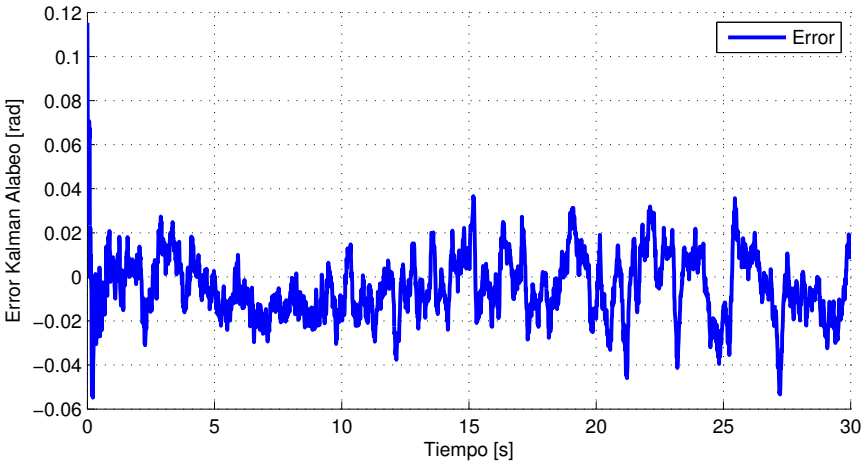

Figura 47: error en la estimación del alabeo del filtro de Kalman

Tabla 13: características de la estimación de la guiñada realizada por el filtro de Kalman

Error cuadrático medio	0,0151 rad
Error medio	-0,0023 rad
Error máximo	0,0872 rad

cabeceo. El sistema comienza con 3 segundos a 0 radianes para ser sometido a un tren de impulsos de amplitud creciente de amplitud 0,2, 0,6 y 1 radianes con un periodo de 8 segundos cada pulso.

Una vez más, como se puede apreciar en las gráficas de las figuras 48 y 49, la mayor desviación se produce durante la inicialización del filtro. En este caso, la amplitud del pulso no afecta al error en la estimación. Si se recuerdan las ecuaciones diferenciales no lineales que rigen el comportamiento en guiñada, se observa cómo las no linealidades son de segundo orden, por lo que su influencia es limitada.

En la tabla 13 se realiza un resumen de las características de la estimación de ángulo de guiñada realizada por el filtro de Kalman.

8.2.4. Respuesta frente a no linealidades

Al estar utilizando un estimador lineal sobre un sistema no lineal linealizado, cuando los términos no linealidades adquieren peso, se está cometiendo un error de aproximación. Para determinar el comportamiento del estimador frente a las no linealidades se repitió la prueba de alabeo, esta vez con pulsos de hasta 0,6 radianes de amplitud, mientras que la guiñada seguía una señal senoidal de amplitud 0,1 radianes con una frecuencia de 2 $^{\mathrm{rad}}/_{\mathrm{s}}$.

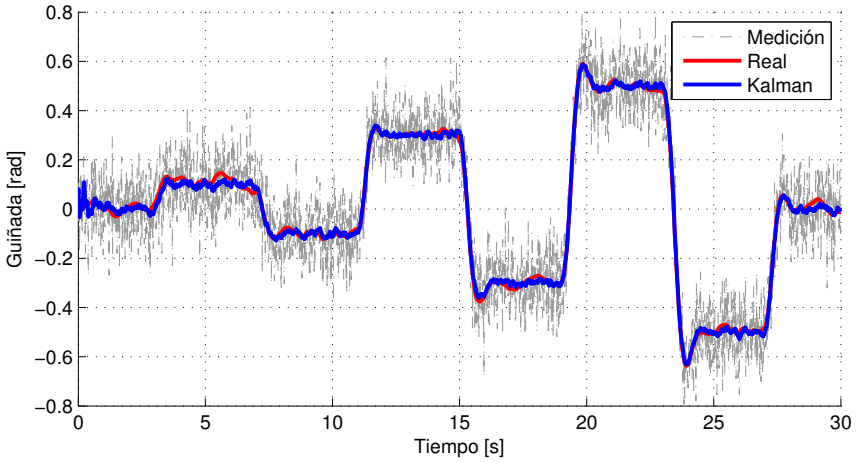

Figura 48: estimación de la guiñada del filtro de Kalman

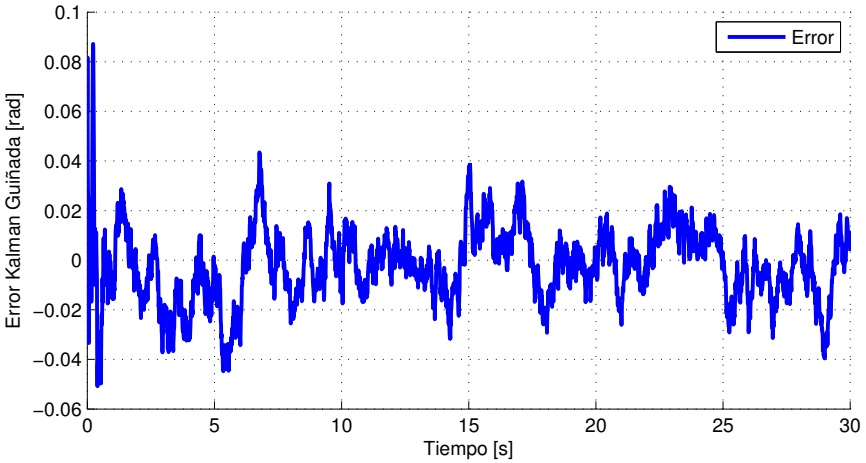

Figura 49: error en la estimación de la guiñada del filtro de Kalman

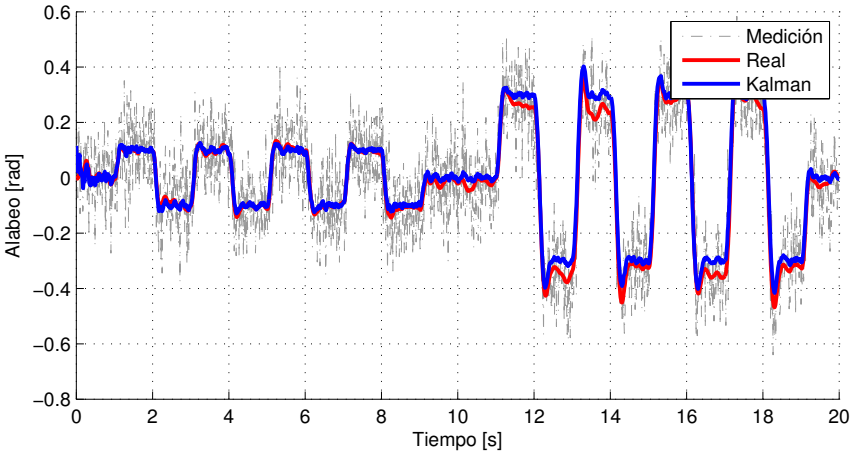

Figura 50: estimación del alabeo del filtro de Kalman cuando aumenta el peso de los términos no lineales

Tabla 14: características de la estimación realizada por el filtro de Kalman cuando aumenta el peso de los términos no lineales

	Alabeo (ϕ)	Guiñada (ψ)
Error cuadrático medio	0,0273 rad	0,0184 rad
Error medio	0,0163 rad	-0,0029 rad
Error máximo	0,1153 rad	0,0872 rad

Como se observa en la gráfica de la figura 50 y en la gráfica del error de la estimación del alabeo de la figura 52, la influencia de los términos no lineales en la estimación del alabeo se hace notable cuando la amplitud de los pulsos aumenta. No ocurre los mismo en la guiñada donde, como se explicó anteriormente, los términos no lineales tienen un peso mucho menor, como se puede ver en las gráficas de las figuras 51 y 53.

En la tabla 14 se resumen las características de la estimación realizada por el filtro de Kalman cuando la influencia de los términos no lineales aumenta. El aumento del error cuadrático medio es notable en la estimación del alabeo, se intuía por la gráfica de la figura 50, mientras que el error cuadrático en la estimación de la guiñada se mantiene prácticamente igual.

8.3. Controlador

8.3.1. Ajuste del controlador

Para realizar el ajuste del controlador PD se optó por usar la simulación con las funciones de transferencia obtenidas anteriormente, ecuaciones (7.71), (7.72) y (7.78)

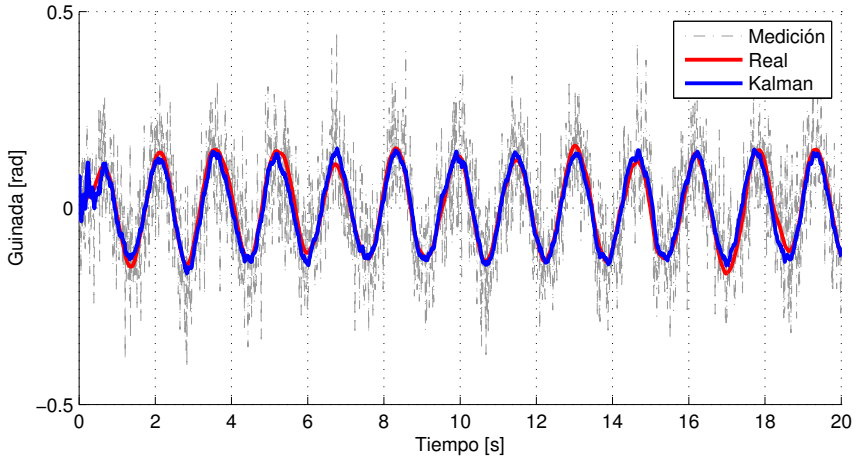

Figura 51: estimación de la guiñada del filtro de Kalman cuando aumenta el peso de los términos no lineales

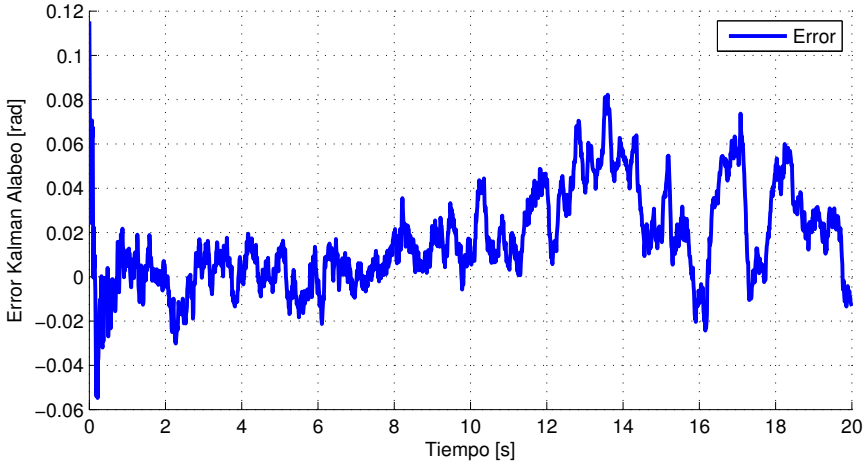

Figura 52: error en la estimación del alabeo del filtro de Kalman cuando aumenta el peso de los términos no lineales

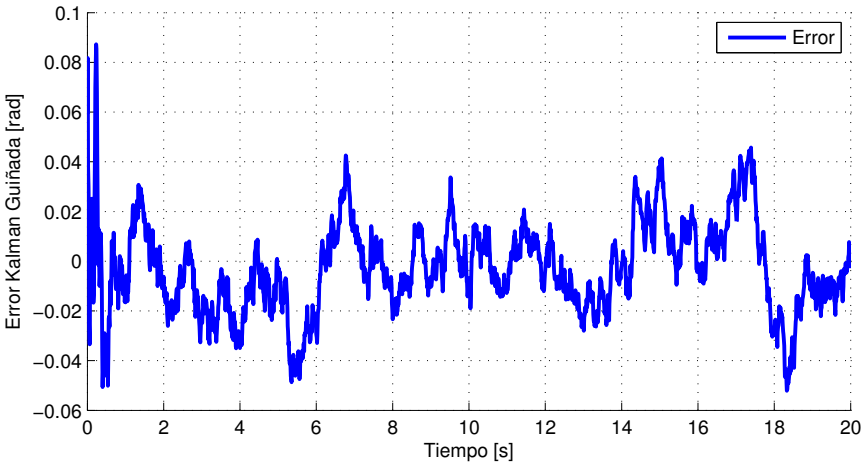

Figura 53: error en la estimación de la guiñada del filtro de Kalman cuando aumenta el peso de los términos no lineales

en lugar de realizar la simulación con modelo dinámico completo que se derivó también previamente. El modelo utilizado es el de la figura (54).

Esta elección se realizo debido a que la velocidad de ejecución de la simulación sobre el modelo dinámico completo es muy lenta y, como se verá a continuación, las funciones de transferencia obtenidas se ajustan bastante bien al proceso.

Por las características del sistema no fue posible ajustar el controlador PD usando uno de los dos métodos de Ziegler-Nichols. Al poseer un doble integrador el primer método no es válido y, como la función de transferencia tiene sólo dos polos en el origen, el sistema oscila constante para cualquier valor de K_p por lo que el segundo método tampoco se podía aplicar. Como consecuencia el ajuste de las ganancias del controlador se hizo de manera manual.

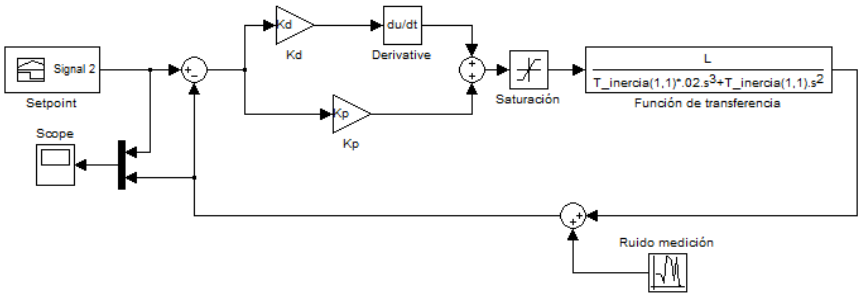

Figura 54: modelo en Simulink del controlador PD usando la función de transferencia

Tabla 15: ganancias de los controladores PD de alabeo, cabeceo y guiñada

	K_p	K_d
Alabeo/Cabeceo	70	5
Guiñada	100	22

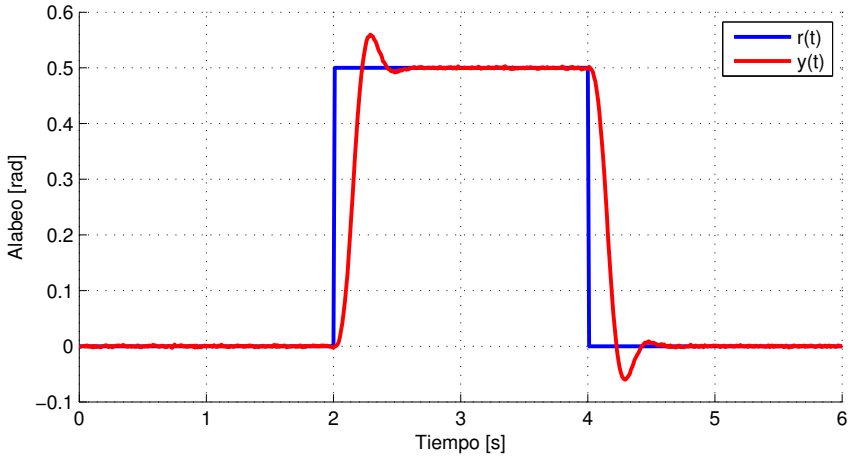

Figura 55: respuesta a entrada pulso del controlador PD usando la función de transferencia

Con el ajuste del controlador se buscó un tiempo de asentamiento lo suficientemente rápido sin que se superase una oscilación del 10 %. Las ganancias del controlador PD de alabeo-cabeceo y del controlador de guiñada se muestran en la tabla 15.

Como se observa en la gráfica de la figura 55 de la respuesta a una entrada pulso de 0,5 radianes y 2 segundos de duración, el sobrepaso es del 11,96 % siendo el tiempo de asentamiento de 0,37 segundos.

8.3.2. Respuesta a escalón

Tras la obtención de unos valores para las ganancias del controlador usando la función de transferencia, se evaluó el rendimiento del controlador usando esta vez el modelo dinámico completo.

Aunque la idea inicial era realizar un ajuste fino del controlador partiendo de los valores obtenidos previamente, tras la realización de los primeros ensayos, se decidió que no era necesario. Como se puede apreciar al comparar los valores de sobrepaso y de tiempo de asentamiento de las gráficas de las figuras 55 y 56, la respuesta del modelo dinámico es prácticamente igual a la de la función de transferencia.

La respuesta del sistema a una entrada de tipo escalón de 0,5 radianes en alabeo y en guiñada se puede ver en las gráficas de las figuras 56 y 57. El sobrepaso y el tiempo de asentamiento del sistema frente a la entrada escalón se encuentra en la tabla 16.

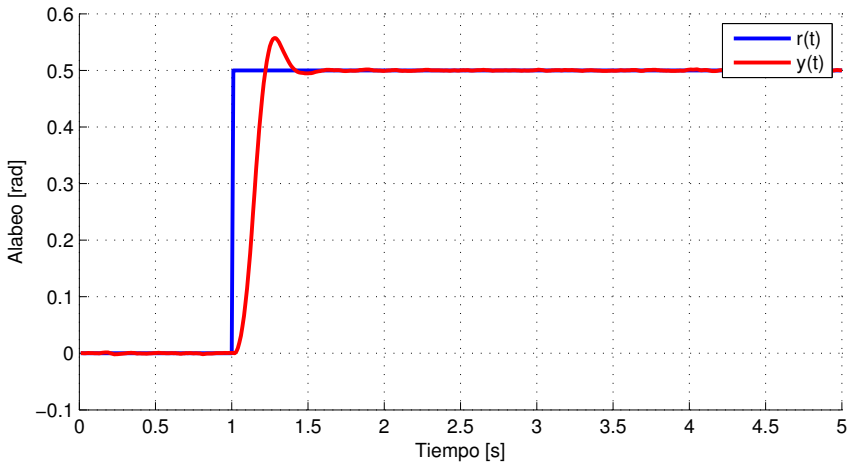

Figura 56: respuesta a entrada escalón en alabeo

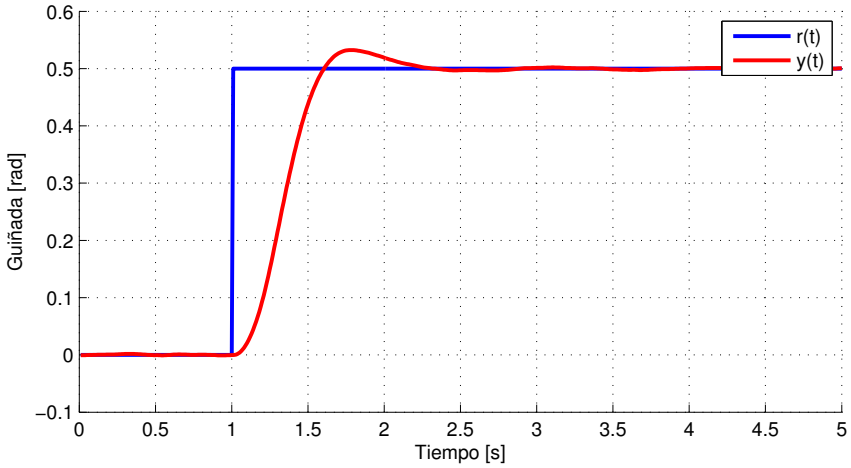

Figura 57: respuesta a entrada escalón en guiñada

Tabla 16: características de la respuesta del controlador PD a entrada escalón de 0,5 radianes

	Sobrepaso	Tiempo de asentamiento
Alabeo/Cabeceo	11,44 %	0,35 s
Guiñada	6,46 %	0,94 s

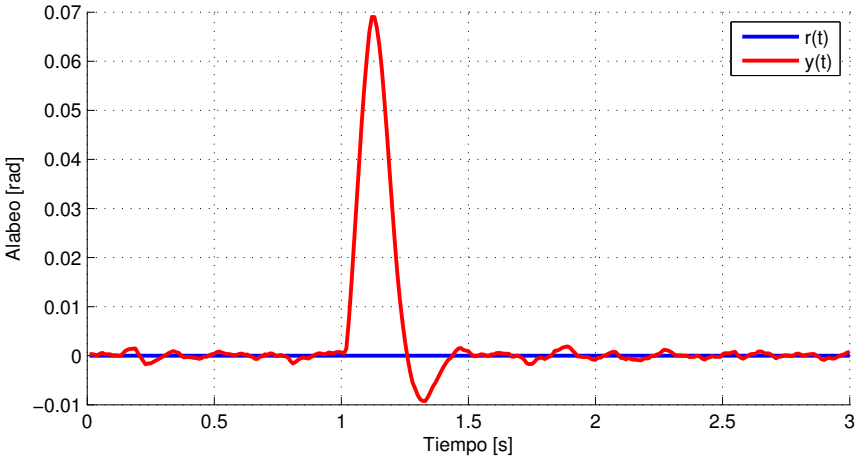

Figura 58: respuesta en alabeo a perturbación de 1N·m durante 0,1 s

8.3.3. Respuesta perturbaciones

Entre los objetivos del proyecto se encontraba que el vuelo del quadrotor fuese estático, autónomo y estable frente a perturbaciones externas. Para comprobar este último punto se le aplicó al quadrotor un par sobre el eje x_b de 1 N·m durante 0,1 segundos. La respuesta del controlador a esta perturbación se puede apreciar en la gráfica de la figura 58, siendo por tanto, una respuesta aceptable para este problema de control.

Desde que el momento deja de aplicarse el sistema tarda 0,34 segundos en volver al punto de consigna, produciéndose la máxima desviación a los 0,13 segundos de haberse aplicado el momento, con un valor de 0,069 radianes.

Parte V

Conclusiones

9. Conclusiones

Sobre el modelo dinámico del quadrotor diseñado en Simulink, se simuló el comportamiento de los sensores y se hicieron correr los algoritmos de la unidad de medición inercial, del filtro de Kalman y del controlador PD.

La unidad de medición inercial estima los ángulos de Tait-Bryan a partir de los sensores. El algoritmo utilizado es el propuesto por Euston *et al.* [14], el cual está diseñado para la estimación de la orientación en aeronaves de ala fija. En el artículo de Euston *et al.,* a la aceleración medida por los acelerómetros se le resta la aceleración centrípeta, mientras que en el algoritmo que se implemento esto no era posible al no tener la referencia externa proporcionada por un GPS o un sensor de flujo óptico, lo que afectó a la calidad de la estimación. Aun así en alabeo el error cuadrático medio obtenido en los ensayos fue de 0,0281 radianes (1,6100°), llegando a un error máximo en la estimación de 0,0659 radianes (3,7758°) cuando el alabeo es de 0,5 radianes (28,6479°).

Con la orientación obtenida por la unidad de medición inercial y el modelo dinámico, el filtro de Kalman realiza una estimación de la orientación. El modelo dinámico del quadrotor se rige por ecuaciones diferenciales no lineales, por lo que al usar un estimador lineal dicho modelo se linealizó respecto al punto de equilibrio. Los efectos de la aproximación realizada al linealizar se hacen notables al combinar el movimiento en alabeo o cabeceo con una rotación en guiñada. En este caso el error cuadrático medio en la estimación del alabeo fue de 0,0273 radianes (1,5641°). Cuando el peso de los términos no lineales no tiene tanta influencia, el error cuadrático medio en la estimación del alabeo es de 0,0154 radianes (0,8824°). Como se puede observar, el error cuadrático medio, aún en condiciones desfavorables, de la estimación realizada por el filtro de Kalman es menor que el de la estimación realizada por la unidad de medición inercial.

Para controlar la orientación se recurrió a tres controladores PD, uno por cada eje de rotación. Los criterios de diseño fueron lograr el menor tiempo de asentamiento sin que el sobrepaso superase el 10 %. Con estos criterios de diseño se ajustaron las ganancias de forma manual. Como resultado, en el controlador de alabe y cabeceo, el tiempo de asentamiento fue de 0,35 segundos siendo el sobrepaso del 11,44 % siendo la entrada una señal escalón de 0,5 radianes (28,6479°) de amplitud. En guiñada el tiempo de asentamiento fue de 0,94 segundos con un sobrepaso del 6,46 %. La salida de los controladores se limitó mediante saturaciones de forma que el empuje realizado por un rotor nunca fuese negativo.

A la hora de implementar el algoritmo de la unidad de medición inercial y del filtro de Kalman el Arduino, la velocidad de ejecución del bucle de control resulta fundamental. Se consiguió ejecutar el bucle completo en 6 milisegundos, lo que significa que se

Figura 59: Parrot AR.Drone 2.0 con protección

efectúa el bucle de control con una frecuencia de 166 Hz, considerándose necesario para lograr un vuelo estable que el bucle de control se ejecute a 100 Hz como mínimo.

10. Proyectos futuros

En el campo del control automático este puede ser el primero de otros muchos proyectos relacionados con vehículos aéreos autónomos del tipo quadrotor. Por ejemplo, se puede comparar el resultado del uso de diferentes técnicas de control como pueden ser el control lineal cuadrático o el control difuso en el control de la orientación del quadrotor, aunque, tras lograr el control en orientación, el siguiente paso sería el control en posición, sin embargo, para ello habría que equipar al quadrotor de sensores de posición, sea bien mediante GPS o sensores de flujo óptico.

El filtro de Kalman lineal ha demostrado un buen desempeño a la hora de estimar la orientación del quadrotor pese a que se trata de un sistema dinámico no lineal. Un posible proyecto sería el diseño de un estimador no lineal como puede ser un filtro de Kalman extendido.

Otra opción es equipar al quadrotor con sensores de distancia gracias a los cuales sería posible implementar un algoritmo que evitase la colisión de este con los objetos del entorno. Fuera del campo del control automático se podría diseñar una protección para las hélices similar a la que utiliza el AR.Drone 2.0 de Parrot (figura 59) o equipar al quadrotor de motores sin escobillas.

Parte VI
Bibliografía

[1] Al-Younes, Y.M.; Al-Jarrah, M.A.; Jhemi, A.A. Linear vs. nonlinear control techniques for a quadrotor vehicle. 2010 7*th International Symposium on Mechatronics and its Applications (ISMA)*. Sharjah, 20-22 de abril de 2010.

[2] Alexis, K. Model predictive quadrotor indoor position control. *19th Mediterranean Conference on Control & Automation (MED)*. Corfu, 20-23 de junio de 2011.

[3] Aracil Santonja, J; Gordillo Álvarez, F. *Regulación Automática*. s.f.

[4] Barrientos, A.; del Cerro, J.; Gutiérrez, P.; San Martín, R.; Martínez, A.; Rossi, C. Vehículos aéreos no tripulados para uso civil. Tecnología y aplicaciones. *Congreso Español de Informática*. Zaragoza, 2007.

[5] W. Beard, Randal. *Quadrotor Dynamics and Control*. Brigham: Briham Young University, 2008.

[6] Bedford, A; Fowler W. *Engineering Mechanics: Dynamics SI Edition*. Exxes: Addison Wesley Longman, 1996. ISBN: 0-201-40341-2.

[7] Berna Ferri, A. *Desarrollo de una plataforma de tiempo real para la implementación de algoritmos de control multivariables: Ampliación al control de orientación de vehículos aéreos*. Valencia: Universidad de Politécnica de Valencia, 29 de septiembre de 2010.

[8] Berstein, Jonathan. *An Overview of MEMS Inertial Sensing Technology.* <http://www.sensorsmag.com/>. 1 de Febrero de 2003. Consultado el 23 de junio de 2012.

[9] Bouabdallah, S.; Noth, A.; Siegwart, R. *PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor.* Lausanne: Swiss Federal Institute of Technology, 11-14 de septiembre de 2007.

[10] Chan, Ai-Ling; Tan, Su-Lim; Kwek, Chu-Lih. Sensor data fusion for attitude stabilization in a low cost Quadrotor system. *2011 IEEE International Symposium on Consumer Electronics.* Singapur, 14-17 de junio de 2011.

[11] V. Cook, Michael. *Flight Dynamics Principles.* 2ª edición. Burlington: Elsevier, 2007. ISBN: 978-0-7506-6927-6.

[12] Criado Rodríguez, *Javier. Guiado por computador de un helicóptero con cuatro rotores.* Almería: Universidad de Almería, 2009.

[13] Etter, W.; Martin, P.; Mangharam, P. *Cooperative Flight Guidance of Autonomous Unmanned Aerial Vehicles*. University of Pennsylvania, 2011.

[14] Euston, Mark; Coote, Paul; Mahony, Robert; Kim, Jonghyuk; Hamel, Tarek. A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV. *2008 IEEE International Conference on Intelligent Robots and Systems (IROS)*. Niza, 22-26 de septiembre de 2008.

[15] García de Marina, Hector; J. Pereda, Fernando; Giron-Sierra, Jose M; Espinosa, Felipe. UAV Attitude Estimation Using Unscented Kalman Filter and TRIAD. *IEEE Transactions on Industrial Electronics*. 11 de noviembre de 2012, vol. 59, núm 11.

[16] S. Grewal, Mohinfer; P. Andrews, Angus. *Kalman Filtering: Theory and Practice Using MATLAB*. 2ª edición. Nueva York: Wiley-Interscience, 2001. ISBN: 0-471-26638-8.

[17] Hoffmann, Frank; Goddemeier, Niklas; Bertram, Tortsen. Attitude estimation and control of a qudcopter. *2010 IEEE International Conference on Intelligent Robots and Systems*. Taipei, 18-22 de octubre de 2010.

[18] Jiménez Gajate, R. *Diseño y desarrollo de un sistema de control para un quadrotor*. Almería: Universidad de Almería, 2012.

[19] Jiong, Yi; Lei, Zhang; Jiangping, Deng; Rong, Shu; Jianyu, Wang. GPS/SINS/BARO Integrated Navigation System for UAV. *2010 International Forum on Information Technology and Application*s *(IFITA)*. Kunming, 16-18 de julio de 2010.

[20] Khebbache, Hicham; Sait, Belkacem; Yacef, Fouad; Soukkou, Yassine. Robust Stabilization of a Quadrotor Aerial Vehicle in Presence of Actuator Faults. *2012 International Journal of Information Technology, Control and Automation (IJITCA)*. Abril 2012, vol. 2, núm. 2.

[21] C. Kuo, B. *Sistemas de Control Automático*. 7ª edición. Naucalpan de Juárez: Prentice-Hall Hispanoamericana, 1996. ISBN: 968-880-723-0.

[22] Lendel, Z.; Berna, A.; Guzmán-Giménez, J.; Sala, A.; García, P. *Application of Takagi-Sugeno observers for state estimation in a quadrotor*. Valencia: Universidad de Politécnica de Valencia, 2011.

[23] Li, J.; Li, Y. Dynamic Analysis and PID Control for a Quadrotor. *IEEE International Conference on Mechatronics and Automation*. Beijing, 7-10 de agosto de 2011.

[24] Lupashin, S.; Schöllig, A.; Sherback, M.; D'Andrea, R. A simple learning strategy for high-speed quadrocopter multi-flips. *2010 IEEE International Conference on Robotics and Automation (ICRA)*. Anchorage, 3-8 de mayo de 2010.

[25] Lupashin, S.; D'Andrea, R. Adaptive Open-Loop Aerobatic Maneuvers for Quadrocopters. *2010 IEEE International Conference on Robotics and Automation (ICRA)*. Anchorage, 3-8 de mayo de 2010.

[26] Martin, Philippe; Salaün, Erwan. The True Role of Accelerometer Feedback in Quadrotor Control. *18th International Federation of Automatic Control World Congress*. Milán, 28 de agosto-2 de septiembre de 2011.

[27] Muñoz Cerezo, J. S. *Desarrollo de un sistema de tiempo real para el control de un mini-helicóptero de 4-rotores en vuelo libre*. Valencia: Universidad de Politécnica de Valencia, 16 de julio de 2010.

[28] Nadales Real, Christian. *Control de un Quadrotor mediante la plataforma Arduino.* Barcelona: Escola Politècnica Superior de Castelldefels, 2009.

[29] Ogata, K. *Ingeniería de Control Moderna*. 4ª edición. Madrid: Pearson Educación, 2003. ISBN: 84-205-3678-4.

[30] Petersen, Christian Fink; Hansen,Henrik; Larsson, Steen; Theilgaard Madsen, Lars Bo; Rimestad, Michael. *Autonomous Hovering with a Quadrotor Helicopter.* Aalborg: Aalborg Universitet, junio de 2008.

[31] V. Raffo, Guilherme; G. Ortega, Manuel; R. Rubio, Francisco. Backstepping/-Nonlinear H_∞ Control for Path Tracking of a QuadRotor Unmanned Aerial Vehicle. *American Control Conference*. Seattle, 11-13 de junio de 2008.

[32] F. Roley, William; D. Sturger, Leroy. *Ingeniería Mecánica: Dinámica*. Madrid: Reverte, 1996. ISBN: 84-291-4256-8.

[33] Simons, M. *Model Aircraft Aerodynamics*. 4ª edición. Dorset: Nexus Special Interests, 1999. ISBN: 18-548-6190-5.

[34] Thompshon Dydek, Zachary. *Adaptative Control of Unmanned Aerial Systems*. Massachesetts: Massachuetts Institute of Technology, septiembre de 2010.

[35] Welch, Greg; Bishop, Gary. *An Introduction to the Kalman Filter*. 2006.

[36] Zhou, Quin-Li; Zhang, Youmin; Qu, Yao-Hong; Rabbath, Camille-Alain. Dead reckoning and Kalman filter design for trajectory tracking of a quadrotor UAV. *2010 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications (MESA)*. Suzhou, 8-10 de julio de 2010.

[37] Analog Devices | Semiconductors. <http://www.analog.com/en/index.html>. Consultado el 25 de junio de 2012.

[38] AR.Drone 2.0. Parrot new Wi-Fi quadricopter. <http://ardrone2.parrot.com/> Consultado el 10 de agosto de 2012.

[39] Arduino - Homepage. <http://www.arduino.cc/>. Consultado el 26 de junio de 2012.

[40] Arduino - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/Arduino>. Consultado el 26 de junio de 2012.

[41] Draganfly.com Industrial Aerial Video Systems & UAVs. <http://www.draganfly.com/>. Consultado el 10 de agosto de 2012.

[42] ETH - IDSC - Flying Machine Arena. <http://www.idsc.ethz.ch/Research_DAndrea/FMA>. Consultado el 10 de agosto de 2012.

[43] Euler Angles - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/Euler_angles>.Consultado el 28 de abril de 2012.

[44] Fuzzy Logic - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/Fuzzy_logic>. Consultado el 26 de julio de 2012.

[45] Honeywell Magnetic Sensors. <http://www.magneticsensors.com/>. Consultado el 25 de junio de 2012.

[46] InvenSense | MotionProcessing | MEMS Gyro | MotionSensing. <http://www.invensense.com/>. Consultado el 25 de junio de 2012.

[47] Kalman filter - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/Kalman_filter>. Consultado el 28 de abril de 2012.

[48] Model Predictive Control - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/Model_predictive_control>.Consultado el 29 de julio de 2012.

[49] MathWorks España - MATLAB - El lenguaje del cálculo técnico. <http://www.mathworks.es/products/matlab/>. Consultado el 28 de agosto de 2012.

[50] MathWorks España - Simulink - Simulación y diseño basado en modelos. <http://www.mathworks.es/products/simulink/>. Consultado el 28 de agosto de 2012.

[51] PID Controler - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/PID_controler>. Consultado el 28 de julio de 2012.

[52] Quadrotor - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/Quadrotor>. Consultado el 12 de agosto de 2012.

[53] Robust Control - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/Robust_control>. Consultado el 28 de julio de 2012.

[54] sf9domahrs - AHRS for Sparkfun's "9DOF Razor IMU". <http://code.google.com/p/sf9domahrs/>. Consultado el 28 de julio de 2012.

[55] Unmanned aerial vehicle - Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle>. Consultado el 12 de agosto de 2012.

Parte VII

Anexos

11. Código fuente Arduino

11.1. Módulo principal

```
/*
 *
 * Quadrotor
 *
 */

#include <Wire.h>
#include <PID_v1.h>
#include <Servo.h>

/************** MACROS **************/
#define toDeg(x) x*57.3
#define toPWM(x) int(x)


/************** CONSTANTES **************/

// Resgistros de los sensores
#define IMU3K 0x68          // gyro I2C address
#define REG_GYRO_X 0x1D    // IMU-3000 Register address for
    GYRO_XOUT_H
#define ACCEL 0x53         // Accel I2c Address
#define ADXL345_POWER_CTL 0x2D
#define HMC5883  0x1E //mag I2C address

// Gravedad
#define G_ACEL      256
#define DT 10

// Ganancias PI DCM
#define Kp_ROLLPITCH_DCM .05
#define Ki_ROLLPITCH_DCM 0.00002

// Ganancias controlador PD
#define Kp_ROLLPITCH 50
#define Kd_ROLLPITCH 10
#define Kp_YAW 10
#define Kd_YAW 2

/************** Variables **************/

char buffer[12];

// Datos de los motores
Servo motor1, motor2, motor3, motor4;

```

```
// Datos de las aceleraciones
float acel[3] = {
  0, 0, 0};
int acelOffset[3] = {
  2, 3, 55};

// Datos de los giros
float giro[3] = {
  0, 0, 0};
int giroOffset[3] = {
  0, 0, 0};

// Angulos de Euler
double pitch, roll, yaw;

// Variables para calcular el dt
float time, ptime;
float dt;

// Contadores
unsigned int count;

// Variable IMU
float DCM[3][3] = {
  {
    1,0,0        }
  ,{
    0,1,0        }
  ,{
    0,0,1        }
};
float UpdateMatrix[3][3] = {
  {
    0,1,2        }
  ,{
    3,4,5        }
  ,{
    6,7,8        }
};
float TempMatrix[3][3] = {
  {
    0,0,0        }
  ,{
    0,0,0        }
  ,{
    0,0,0        }
};

float Omega_P[3]= {
  0,0,0};
float Omega_I[3]= {
  0,0,0};
float Omega[3]= {
  0,0,0};

float errorRollPitch[3]= {
  0,0,0};
float errorYaw[3]= {
```

```
  0,0,0};

//Variables Kalman
float A[2][2] = {
  {
    1,DT         }
  ,{
    0,1          }
};
//float B[3][2] = {{0,DT*K_RP/I[0]},{0, DT*K_RP/I[1]},{0, DT*
    K_YAW/I[2]}};  <-- *********CORREGIR**********
float B[3][2] = {
  {
    0,DT*1          }
  ,{
    0, DT*1          }
  ,{
    0, DT*1          }
};
float H[2][2] = {
  {
    1,0          }
  , {
    0,1          }
};
float P[3][2][2] = {
  {
    {
      1,0                   }
    ,{
      0,1                   }
  }
  ,{
    {
      1,0                   }
    ,{
      0,1                   }
  }
  ,{
    {
      1,0                   }
    ,{
      0,1                   }
  }
};
float Q[3][2][2] = {
  {
    {
      .1,0                   }
    ,{
      0,.1                   }
  }
  ,{
    {
      .1,0                   }
    ,{
      0,.1                   }
  }
```

```
  ,{
    {
      .1,0                  }
    ,{
      0,.1                  }
  }
};
float R[3][2][2] = {
  {
    {
      .0001,0                   }
    ,{
      0,.0001                   }
  }
  ,{
    {
      .0001,0                   }
    ,{
      0,.0001                   }
  }
  ,{
    {
      .0001,0                   }
    ,{
      0,.0001                   }
  }
};
float K[3][2][2] = {
  {
    {
      1,0                   }
    ,{
      0,1                   }
  }
  ,{
    {
      1,0                   }
    ,{
      0,1                   }
  }
  ,{
    {
      1,0                   }
    ,{
      0,1                   }
  }
};

// Variable de los controladores PD
double setpoint[3] = {
  0,0,0};
double PWM_Offset, PWM_roll, PWM_pitch, PWM_yaw;
float PWM_Motor[4] = {
  0,0,0,0};

PID rollPD(&roll, &PWM_roll, &setpoint[0], Kp_ROLLPITCH, 0.0,
    Kd_ROLLPITCH, DIRECT);
```

```
217 PID pitchPD(&pitch, &PWM_pitch, &setpoint[1], Kp_ROLLPITCH, 0,
        Kd_ROLLPITCH, DIRECT);
218 PID yawPD(&yaw, &PWM_yaw, &setpoint[2], Kp_YAW, 0, Kd_YAW,
        DIRECT);
219
220 /************** SETUP y LOOP **************/
221
222 void setup() {
223
224   pinMode(13, OUTPUT);
225
226   Serial.begin(57600);
227
228   Wire.begin();
229   // Sample Rate 1kHz, Filter Bandwidth 42Hz, Gyro Range 2000
          d/s
230   writeTo(IMU3K, 0x16, 0x1B);
231   //set accel register data address
232   writeTo(IMU3K, 0x18, 0x32);
233   // set accel i2c slave address
234   writeTo(IMU3K, 0x14, ACCEL);
235
236
237   // Set passthrough mode to Accel so we can turn it on
238   writeTo(IMU3K, 0x3D, 0x08);
239   // Turn on accel
240   writeTo(ACCEL, ADXL345_POWER_CTL, 0);
241   writeTo(ACCEL, ADXL345_POWER_CTL, 16);
242   writeTo(ACCEL, ADXL345_POWER_CTL, 8);
243   //Set accel range 2g, full resolution(10bit) and MSB
244   writeTo(ACCEL, 0x31, 0x08);
245   //cancel pass through to accel, gyro will now read accel for
          us
246   writeTo(IMU3K, 0x3D, 0x28);
247
248   delay(1000);
249
250   // Corrijo la deriva de los giroscopios
251   zeroGyro();
252
253   // Inicio los controladores PD
254   rollPD.SetMode(AUTOMATIC);
255   pitchPD.SetMode(AUTOMATIC);
256   yawPD.SetMode(AUTOMATIC);
257
258   // Limito la salida de los PD
259   pitchPD.SetOutputLimits(-20, 20);
260   yawPD.SetOutputLimits(-20, 20);
261   rollPD.SetOutputLimits(-50, 50);
262
263   motor1.attach(10);
264   motor2.attach(5);
265   motor3.attach(11);
266   motor4.attach(6);
267
268   motor1.write(200);
269   motor3.write(200);
270
```

```
271   delay(3000);
272
273   motor1.write(30);
274   motor3.write(30);
275
276   delay(3000);
277
278   PWM_Offset = 50;
279 }
280
281 void loop() {
282
283   if (float(millis() - ptime) >= DT) {
284
285     dt = float(millis() - ptime)/1000;
286     ptime = millis();
287
288     ReadAccelGyro();
289
290     // IMU
291     MatrixUpdate();
292     Normalize();
293     DriftCorrection();
294     Euler();
295
296     // Kalman
297     Kalman();
298
299     // PID
300     rollPD.Compute();
301     pitchPD.Compute();
302     yawPD.Compute();
303
304     if (Serial.available() <= 0) {
305       PWM_Motor[0] = PWM_Offset + PWM_pitch;// - PWM_yaw;
306       PWM_Motor[1] = PWM_Offset - PWM_roll;// + PWM_yaw;
307       PWM_Motor[2] = PWM_Offset - PWM_pitch;// - PWM_yaw;
308       PWM_Motor[3] = PWM_Offset + PWM_roll;// + PWM_yaw;
309     }
310     else {
311       PWM_Motor[0] = 30;
312       PWM_Motor[1] = 30;// + PWM_yaw;
313       PWM_Motor[2] = 30;// - PWM_yaw;
314       PWM_Motor[3] = 30;// + PWM_yaw;
315     }
316
317     motor1.write(PWM_Motor[0]);
318     motor3.write(PWM_Motor[2]);
319
320     for (int i=0; i < 4; i++) {
321       Serial.print(toPWM(PWM_Motor[i]));
322       Serial.print(",");
323     }
324
325     /*Print output
326      Serial.print(toDeg(roll));
327      Serial.print(",");
328      Serial.print(toDeg(pitch));
```

```
329       Serial.print(",");
330       Serial.print(toDeg(yaw));
331       Serial.print(",");*/
332      Serial.print(dt*1000);
333      Serial.println("");
334    }
335 }
```

Listado de programa 4: modulo principal

11.2. Librería de cálculo matricial

```
1  void Matriz3x3_Producto(float out[3][3], float a[3][3], float
       b[3][3]) {
2
3          float op[3];
4
5          for(int i = 0; i < 3; i++) {
6                  for(int j = 0; j < 3; j++) {
7                          for(int k = 0; k < 3; k++) {
8                                  op[k] = a[i][k]*b[k][j];
9                          }
10                         out[i][j] = op[0]+op[1]+op[2];
11                 }
12         }
13 }
14
15 void Matriz2x2_Producto(float out[2][2], float a[2][2], float
       b[2][2]) {
16
17         float op[2];
18
19         for(int i = 0; i < 2; i++) {
20                 for(int j = 0; j < 2; j++) {
21                         for(int k = 0; k < 2; k++) {
22                                 op[k] = a[i][k]*b[k][j];
23                         }
24                         out[i][j] = op[0]+op[1];
25                 }
26         }
27 }
28
29 void Matriz2x2_Suma(float out[2][2], float a[2][2], float b
       [2][2]) {
30         out[0][0]= a[0][0] + b[0][0];
31         out[0][1]= a[0][1] + b[0][1];
32         out[1][0]= a[1][0] + b[1][0];
33         out[1][1]= a[1][1] + b[1][1];
34 }
35
36 void Matriz2x2_Trans(float out[2][2], float a[2][2]) {
37         out[0][0]= a[1][1];
38         out[0][1]= a[1][0];
39         out[1][0]= a[0][1];
40         out[1][1]= a[0][0];
41 }
```

```
42
43 void Matriz2x2_Inv(float out[2][2], float a[2][2]) {
44
45         float det = a[0][0]*a[1][1] - a[0][1]*a[1][0];
46
47         out[0][0]=  a[1][1]/det;
48         out[0][1]= -a[1][0]/det;
49         out[1][0]= -a[0][1]/det;
50         out[1][1]=  a[0][0]/det;
51 }
```

Listado de programa 5: librería para el cálculo matricial

11.3. Librería de cálculo vectorial

```
1  void Vector_PorEscalar(float out[3], float a, float b[3]) { //
       [out] = a * [b]
2          for(int i = 0; i<3; i++) {
3                  out[i] = a*b[i];
4          }
5  }
6
7  void Vector_Suma(float out[3], float a[3], float b[3]) { // [
       out] = [a] + [b]
8          for(int i = 0; i<3; i++) {
9                  out[i] = a[i]+b[i];
10         }
11 }
12
13 void Vector_Cross(float out[3], float a[3], float b[3]) { //
       out = [a] x [b]
14         out[0]= (a[1]*b[2]) - (a[2]*b[1]);
15         out[1]= (a[2]*b[0]) - (a[0]*b[2]);
16         out[2]= (a[0]*b[1]) - (a[1]*b[0]);
17 }
18
19 float Vector_Dot(float a[3], float b[3]) { // [out] = [a] Â· [
       b]
20         float out = 0;
21         for(int i = 0; i < 3; i++) {
22                 out += a[i]*b[i];
23         }
24         return out;
25 }
```

Listado de programa 6: librería para el cálculo vectorial

11.4. Unidad de medición inercial

```
1 // Leo datos del acelerometro y del giroscopio
2 void ReadAccelGyro() {
3   int i = 0;
4
```

```
  Wire.beginTransmission(IMU3K);
  Wire.write(REG_GYRO_X); //Register Address GYRO_XOUT_H
  Wire.endTransmission();

  Wire.beginTransmission(IMU3K);
  Wire.requestFrom(IMU3K,12); // Read 12 bytes
  while(Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();

  // El formato del giro es MSB primero
  giro[1] = -((buffer[0] << 8 | byte(buffer[1]))-giroOffset
      [0]);
  giro[0] = -((buffer[2] << 8 | byte(buffer[3]))-giroOffset
      [1]);
  giro[2] = -((buffer[4] << 8 | byte(buffer[5]))-giroOffset
      [2]);

  // El formato del acelerometro es LSB primero
  acel[1] = -(buffer[7] << 8 | byte(buffer[6]))+acelOffset[1];
  acel[0] = -(buffer[9] << 8 | byte(buffer[8]))+acelOffset[0];
  acel[2] = (buffer[11] << 8 | byte(buffer[10]))+acelOffset
      [2];

  //Devuelvo el resultado del giro en rad/s
  Vector_PorEscalar(&giro[0], .00106, &giro[0]);
}

// Corrijo la desviacion de los giros
void zeroGyro() {

  int i;

  Wire.beginTransmission(IMU3K);
  Wire.write(REG_GYRO_X); //Register Address GYRO_XOUT_H
  Wire.endTransmission();

  Wire.beginTransmission(IMU3K);
  Wire.requestFrom(IMU3K,12); // Read 12 bytes
  while(Wire.available()) {
    buffer[i] = Wire.read();
    i++;
  }
  Wire.endTransmission();

  giroOffset[0] = (buffer[0] << 8 | byte(buffer[1]));
  giroOffset[1] = (buffer[2] << 8 | byte(buffer[3]));
  giroOffset[2] = (buffer[4] << 8 | byte(buffer[5]));
}

// Actualizo matriz R
void MatrixUpdate() {

        // Sumo el termino proporcional e integral del
            controlador
```

```
58      Vector_Suma(&Omega[0], &giro[0], &Omega_I[0]);
59      Vector_Suma(&Omega[0], &Omega[0], &Omega_P[0]);
60
61          UpdateMatrix[0][0] =  1;
62          UpdateMatrix[0][1] = -dt*Omega[2];              // -Z
63          UpdateMatrix[0][2] =  dt*Omega[1];              //  Y
64          UpdateMatrix[1][0] =  dt*Omega[2];              //  Z
65          UpdateMatrix[1][1] =  1;
66          UpdateMatrix[1][2] = -dt*Omega[0];              // -X
67          UpdateMatrix[2][0] = -dt*Omega[1];              // -Y
68          UpdateMatrix[2][1] =  dt*Omega[0];              //  x
69          UpdateMatrix[2][2] =  1;
70
71      for(int x=0; x<3; x++) {
72         for(int y=0; y<3; y++) {
73            TempMatrix[x][y] = DCM[x][y];
74         }
75      }
76
77      Matriz_Producto(DCM,TempMatrix,UpdateMatrix);
78  }
79
80  // Normalizo y corrijo la ortogonalidad
81  void Normalize() {
82
83      float error = 0;
84      float temp[3][3] = {{0,0,0},{0,0,0},{0,0,0}};
85      float renorm = 0;
86
87          // Calculo el error
88          error = -Vector_Dot(&DCM[0][0], &DCM[1][0])*0.5;
89
90          // Corrijo errores de ortogonalidad en X
91          Vector_PorEscalar(&temp[0][0], error, &DCM[1][0]);
92          Vector_Suma(&temp[0][0], &DCM[0][0], &temp[0][0]);
93
94          // Corrijo errores de ortogonalidad en Y
95          Vector_PorEscalar(&temp[1][0], error, &DCM[0][0]);
96          Vector_Suma(&temp[1][0], &DCM[1][0], &temp[1][0]);
97
98          // Corrijo errores de ortogonalidad en Z
99          Vector_Cross(&temp[2][0], &temp[0][0], &temp[1][0]);
100
101         // Normalizo el vector X
102         renorm= .5 * sqrt(sq(temp[0][0])+sq(temp[0][1])+sq(
                temp[0][2]));
103         Vector_PorEscalar(&DCM[0][0], 1/renorm, &temp[0][0]);
104
105         // Normalizo el vector Y
106         renorm= .5 * sqrt(sq(temp[1][0])+sq(temp[1][1])+sq(
                temp[1][2]));
107         Vector_PorEscalar(&DCM[1][0], 1/renorm, &temp[1][0]);
108
109         // Normalizo el vector Z
110         renorm= .5 * sqrt(sq(temp[2][0])+sq(temp[2][1])+sq(
                temp[2][2]));
111         Vector_PorEscalar(&DCM[2][0], 1/renorm, &temp[2][0]);
112 }
```

```
113
114 void DriftCorrection(void) {
115
116   //Compensacion de la deriva
117   static float Scaled_Omega_I[3];
118   float acelModulo;
119   float acelPeso;
120
121   //***** Alabeo y Cabeceo ***************
122
123   // Calculo el modulo del vector aceleracion
124   acelModulo = sqrt(Vector_Dot(&acel[0],&acel[0]));
125   acelModulo = acelModulo / G_ACEL; // Scale to gravity.
126
127   // Determino el peso de la informacion sobre la
128   // aceleracion (<0.5G = 0.0, 1G = 1.0 , >1.5G = 0.0)
129   acelPeso = constrain(1 - 2*abs(1 - acelModulo),0,1);
130
131   // Paso la aceleracion a m/s
132   Vector_PorEscalar(&acel[0], .03832, &acel[0]);
133
134   // Calculo el error el alabeo y cabeceo
135   Vector_Cross(&errorRollPitch[0],&acel[0],&DCM[2][0]);
136
137   // Control proporcional
138   Vector_PorEscalar(&Omega_P[0], Kp_ROLLPITCH*acelPeso, &
          errorRollPitch[0]);
139
140   // Control integral
141   Vector_PorEscalar(&Scaled_Omega_I[0], Ki_ROLLPITCH*acelPeso,
           &errorRollPitch[0]);
142   Vector_Suma(Omega_I,Omega_I,Scaled_Omega_I);
143 }
144
145 // Calculo los angulos de Euler
146 void Euler(void)
147 {
148         roll     =  atan2(DCM[2][1],DCM[2][2]);
149         pitch    =  -asin(DCM[2][0]);
150         yaw      =  atan2(DCM[1][0],DCM[0][0]);
151 }
```

Listado de programa 7: algoritmo de la unidad de medición inercial

11.5. Filtro de Kalman

```
1  void Kalman(void) {
2
3         //Matrices
4         float x[3] = {roll, pitch, yaw};
5         float x_dot[3] = {roll, pitch, yaw};
6         float u_input[3] = {PWM_roll, PWM_pitch, PWM_yaw};
7         float z[3] = {roll, pitch, yaw};
8         float Temp2x2[2][2][2];
9         float Cov[2][2];
10
```

```
11          for (int i = 0; i < 3; i++) {
12                  //Estimacion estado a priori
13                  x[i] = x[i] + A[0][1]*x_dot[i];
14                  x_dot[i] = x_dot[i] + B[i][1]*u_input[i];
15
16                  // Estimacion a priori P
17                  Matriz2x2_Trans(Temp2x2[1],A);
18                  Matriz2x2_Producto(Temp2x2[0],A,P[i]);
19                  Matriz2x2_Producto(P[i],Temp2x2[0],Temp2x2[1])
                        ;
20                  Matriz2x2_Suma(P[i],P[i],Q[i]);
21
22                  // Ganancia de Kalman
23                  Matriz2x2_Suma(Temp2x2[0],H,R[i]);
24                  Matriz2x2_Inv(Cov,Temp2x2[0]);
25                  Matriz2x2_Producto(K[i],P[i],Cov);
26
27                  // Estimacion estado a posteriori
28                  x[i] = x[i] + K[i][0][0]*(z[i]-x[i]);
29
30                  // Estimacion P a posteriori
31                  Temp2x2[0][0][0] = 1 - K[i][0][0];
32                  Temp2x2[0][0][1] = -K[i][0][1];
33                  Temp2x2[0][1][0] = -K[i][1][0];
34                  Temp2x2[0][1][1] = 1 - K[i][1][1];
35
36                  Temp2x2[1][0][0] = P[i][0][0];
37                  Temp2x2[1][0][1] = P[i][0][1];
38                  Temp2x2[1][1][0] = P[i][1][0];
39                  Temp2x2[1][1][1] = P[i][1][1];
40
41                  Matriz2x2_Producto(P[i],Temp2x2[0],Temp2x2[1])
                        ;
42          }
43
44          //roll = x[0];
45          //pitch = x[1];
46          //yaw = x[2];
47  }
```

Listado de programa 8: algoritmo del filtro de Kalman